MH1902T

芯片数据手册

兆讯恒达微电子技术（北京）有限公司

修订记录

日期	修订版本	描述	作者
2019-11-20	1.00	初稿完成	MEGAHUNT
2019-12-21	1.10	修改 ADC 通道 0 内部分压电阻值	MEGAHUNT
2020-03-24	1.20	更新 QFN88 管脚定义	MEGAHUNT
2020-06-10	1.30	补充 VBAT33 以及 CHARGE_VCC 电气参数	MEGAHUNT

目　录

图索引

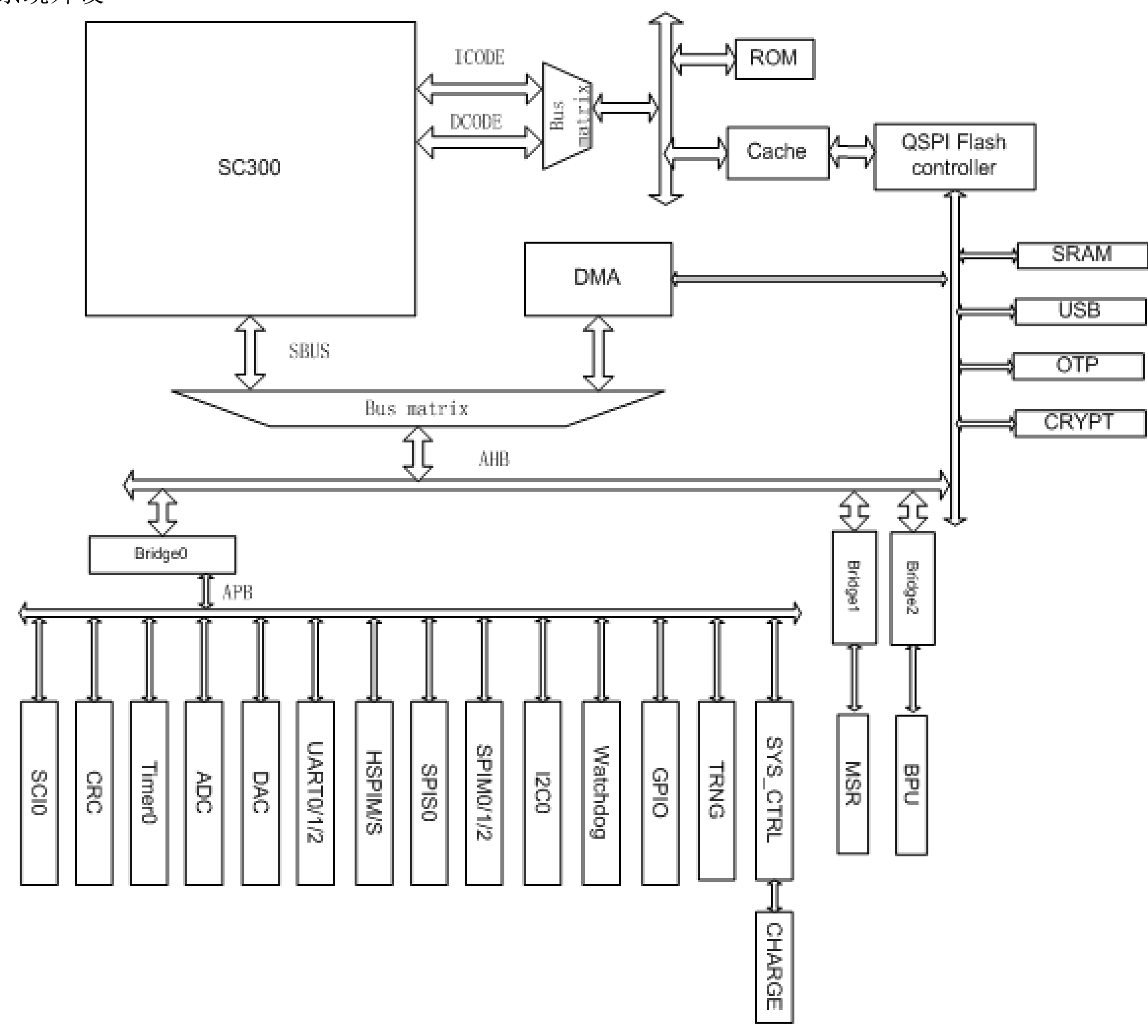
表索引

1 存储器和总线架构

1.1 系统架构

芯片系统有以下单元组成：
系统部分：

- 安全处理器
- 8KB I&D Cache
- 128KB/64KB SRAM
- 系统外设



-

图 1-1 系统架构图

1.2 存储器映射

芯片Flash大小有两种：512KB/1MB，前4KB由内部使用，剩余用于存储固件程序。128Bytes OTP用于存储用户数据。

1.2.1 外设地址映射

下表为SCPU总体地址映射表，外设地址映射见相应章节。

表 1-1 外设地址映射表

地址范围	外设	总线
0x4000_0000-0x4000_03FF	SSC	AHB
0x4000_0800-0x4000_0BFF	DMA	
0x4000_0C00-0x4000_0FFF	USB	
0x4000_1400-0x4000_17FF	HSPI	
0x4000_8000-0x4000_BFFF	OTP	
0x4008_0000-0x4008_FFFF	CACHE	
0x4005_0000-0x4005_FFFF	QSPI	
0x4000_3000-0x4000_6FFF	安全协处理器	
0x4001_0000-0x4001_0FFF	SCI0	APB0
0x4001_1000-0x4001_1FFF	I2C0	
0x4001_2000-0x4001_2FFF	CRC	
0x4001_3000-0x4001_3FFF	Timer0	
0x4001_4000-0x4001_4FFF	ADC	
0x4001_5000-0x4001_5FFF	UART2	
0x4001_6000-0x4001_6FFF	UART0	
0x4001_7000-0x4001_7FFF	UART1	
0x4001_8000-0x4001_8FFF	SPIM1	
0x4001_9000-0x4001_9FFF	SPIM2	
0x4001_A000-0x4001_AFFF	SPIM0	
0x4001_B000-0x4001_BFFF	SPIS0	
0x4001_C000-0x4001_CFFF	Watchdog	
0x4001_D000-0x4001_DFFF	GPIO	
0x4001_E000-0x4001_EFFF	TRNG	
0x4001_F000-0x4001_FFFF	SYSCTRL	
0x4002_0000-0x4002_FFFF	MSR	APB1
0x4003_0000-0x4003_7FFF	BPU	APB2

1.2.2 嵌入式RAM

SCPU内置128KB/64KB的静态SRAM。SRAM可以以字节、半字（16位）或字（32位）访问。SRAM的起始地址：0x2000_0000

外设	地址范围	容量
SRAM	0x2000_0000-0x2001_FFFF	128KB

1.2.3 位段访问

存储器映像包括两个位段(bit-band)区。这两个位段区将别名存储器区中的每个字映射到位段存储器区的一个位，在别名存储区写入一个字具有对位段区的目标位执行读-改-写操作的相同效果。

外设寄存器和SRAM都被映射到一个位段区里，这允许执行单一的位段的写和读操作。

下面的映射公式给出了别名区中的每个字是如何对应位带区的相应位的：

bit_word_addr = bit_band_base + (byte_offset×32) + (bit_number×4)

其中：

bit_word_addr是别名存储器区中字的地址，它映射到某个目标位。

bit_band_base是别名区的起始地址。

byte_offset是包含目标位的字节在位段里的序号

bit_number是目标位所在位置(0-31)

例子：

下面的例子说明如何映射别名区中SRAM地址为0x20000300的字节中的位2：

0x22006008 = 0x22000000 + (0x300×32) + (2×4).

对0x22006008地址的写操作与对SRAM中地址0x20000300字节的位2执行读-改-写操作有着相同的效果。

读0x22006008地址返回SRAM中地址0x20000300字节的位2的值(0x01 或 0x00)。

2 芯片特性说明

2.1 电气特性

表 2-1 极限参数

参数	说明	范围	单位
VDD	稳态电源电压	-0.3 to 3.6	V
Iddpd	关机电流	--	nA
Tamb	工作温度	-40~+80	℃
Tstg	储藏温度	-40~+125	℃
Ground	地	-0.3~0.3	V
Voh	数字输出高电平	VDD -0.3 ~ VDD+0.3	V
Vol	数字输出低电平	<0.4	V
Ioh	拉电流	20	mA
Iol	灌电流	20	mA
Vih	数字输入高电平	⩾0.7×VDD	V
ViL	数字输入低电平	⩽0.3×VDD	V

表 2-2 电气特性

参数	条件（-40°C to +85°C）	值		单位
		最小	最大	
AVD33		2.7	3.6	V
VDD33		2.7	3.6	V
VBAT33		1.9	3.6	V
CHARGE_VCC		4.6	5.5	V
Vol	VDD=3.3V	-	0.4	V
Voh	VDD=3.3V	VDD – 0.3	-	V
VIH	VDD=3.3V	0.7×VDD	-	V
VIL	VDD=3.3V	-	0.3×VDD	V

表 2-3 安全相关特性

传感器	说明	范围	单位
温度传感器	高温检测范围	90~120	℃
	低温检测范围	-30 ~ -50	℃
电压传感器	主电源电压高压检测范围	4.2±0.1	V
	主电源电压低压检测范围	2.7~2.9	V
	电池电压高压检测范围	4.2±0.1	V
	电池电压低压检测范围	2.1±0.1	V
时钟频率传感器	12M时钟频率检测范围	12±50%	MHz
	32K时钟频率检测范围	32±50%	KHz
外部Tamper电阻	Tamper管脚上的上拉电阻阻值	3.8M±10%	Ω

表 2-4 功耗列表

工作模式	说明	功耗	单位
RUN	● 所有外设全开 ■ @144MHz ● 所有外设全关 ■ @144MHz	42 22	mA
CPU Sleep	所有外设全关 @144MHz	7.2	mA
	● 支持 IO 低电平、RTC、攻击、充电和刷卡唤醒	160	uA
VBAT	测试时所有Tamper输入管脚上拉到电源。 ● 主电源掉电 ■ 内部传感器全开 ■ 内部传感器全关 主电源带电且AVD33 – VBAT > 0.3V，内部传感器全开	1.8 1.6 42	uA

表 2-5 充电模块参数

编程电阻值	涓流充电电流(mA)	涓流阈值(V)	恒流充电电流(mA)	充电电压(V)
1K	20±1	2.87±0.01	193±5	4.15±.0.05
2K	10±1	2.88±0.01	96±2	4.15±.0.05
4.02K	5±1	2.9±0.01	46.5±1.5	4.15±.0.05

2.2 管脚定义

2.2.1 QFN68

表 2-6 MH1902T QFN68 8mm×8mm封装引脚定义

PIN No.	PIN name	ALT0	ALT1 (default)	ALT2	ALT3	备注
1	PA4		PA[4]	PWM4	XTAL32K	
2	PA5		PA[5]	PWM5	27.12M	可配置输出27.12M
3	PA8	SCI0_CLK	PA[8]			复用为IO时必须先打开IC卡电源，且输出信号的高电平为IC卡输出电平
4	PA9	SCI0_RSTN	PA[9]			
5	PA10	SCI0_IO	PA[10]			
6	CVCC					IC卡电源
7	VDD33					
8	POWER_KEY					开关机按键
9	EN_LDO5V					外部LDO使能
10	**VSS**					芯片地
11	PA6	SCI0_DET	PA[6]			
12	PA13		PA[13]	CLK_27P12	UART0_RTS	可配置输出27.12M
13	PA15		PA[15]	UART2_TX	UART0_TX	
14	PA0	UART0_RX/IrDA_IN	PA[0]	PWM0		
15	PA1	UART0_TX/IrDA_OUT	PA[1]	PWM1		
16	PA2	UART0_CTS	PA[2]	PWM2		
17	PA3	UART0_RTS	PA[3]	PWM3		
18	PB0	I2C0_SCL	PB[0]	PWM0	XTAL32K	

19	PB1	I2C0_SDA	PB[1]	PWM1	CLK_27P12	
20	PD4	UART1_RX/IrDA_IN	PD[4]			
21	PD5	UART1_TX/IrDA_OUT	PD[5]			
22	PB2	SPI0_SCK	PB[2]	PWM2	SPI3_CLK	
23	PB3	SPI0_CSN0	PB[3]	PWM3	SPI3_CSN0	
24	PB4	SPI0_MOSI	PB[4]	PWM4	SPI3_MOSI	
25	PB5	SPI0_MISO	PB[5]	PWM5	SPI3_MISO	
26	VDD33					IO电源
27	PB12	SPI2_CLK	PB[12]	SPI3_CLK	UART1_RX/IrDA_IN	
28	PB13	SPI2_CSN	PB[13]	SPI3_CSN0	UART1_TX/IrDA_OUT	
29	PB14	SPI2_MOSI	PB[14]	SPI3_MOSI	UART1_CTS	
30	PB15	SPI2_MISO	PB[15]	SPI3_MISO	UART1_RTS	
31	VDD12					
32	RST_N					
33	PC9	UART2_CTS	PC[9]		SPI1_MOSI	
34	PC10	UART2_RX/IrDA_IN	PC[10]		SPI1_CLK	
35	PC15	SPI1_MISO	PC[15]	SPI3_MISO	UART2_TX/IrDA_OUT	
36	PC14	SPI1_MOSI	PC[14]	SPI3_MOSI	UART2_RTS	
37	PC13	SPI1_CLK	PC[13]	SPI3_CLK	UART2_RX/IrDA_IN	
38	PC12	SPI1_CSN	PC[12]	SPI3_CSN	UART2_CTS	
39	CHARGE_VPROG					CHARGE充电电流控制管脚
40	CHARGE_VCC					CHARGE电源输入
41	CHARGE_VBAT					CHARGE电源输出，接电池
42	MSR_IN1	UART1_CTS	PD[6]	MSR_IN1	I2C0_SCL	磁条卡 T1+
43	MSR_IN2	UART1_RTS	PD[7]	MSR_IN2	I2C0_SDA	磁条卡 T1-
44	MSR_IN3	SPI1_SCK	PD[8]	MSR_IN3		磁条卡 T2+
45	MSR_IN4	SPI1_CSN	PD[9]	MSR_IN4		磁条卡 T2-
46	MSR_IN5	SPI1_MOSI	PD[10]	MSR_IN5		磁条卡 T3+
47	MSR_IN6	SPI1_MISO	PD[11]	MSR_IN6		磁条卡 T3-
48	PC5		PC[5]	ADC_IN5		
49	PC4	TCK	PC[4]	ADC_IN4		
50	PC3	TMS	PC[3]	ADC_IN3	UART1_RTS	
51	PC2	TDO	PC[2]	ADC_IN2	UART1_CTS	
52	PC1	TDI	PC[1]	ADC_IN1/DAC	UART1_TX/IrDA_OUT	
53	PC0	TRST	PC[0]	ADC_IN0	UART1_RX/IrDA_IN	
54	DN					USB DN
55	DP					USB DP
56	VBUS					USB VBUS
57	ID					USB ID
58	XI12M					XTAL 12MHz Input
59	XO12M					XTAL12MHz Output
60	VDD33					IO电源
61	AVD33					模拟电源
62	XO32					XTAL 32KHz Output
63	XI32					XTAL 32KHz Input

64	EXTS3					外部Tamper3
65	EXTS1					外部Tamper1
66	EXTS2					外部Tamper2
67	EXTS0					外部Tamper0
68	VBAT33					电池供电电源

注意：

QFN68封装配置Tamper时软件不要关闭内部4路Tamper的上拉或下拉电阻，避免Tamper管脚悬空导致功耗异常

2.2.2 QFN88

表 2-7 MH1902T QFN88 10mm×10mm封装引脚定义

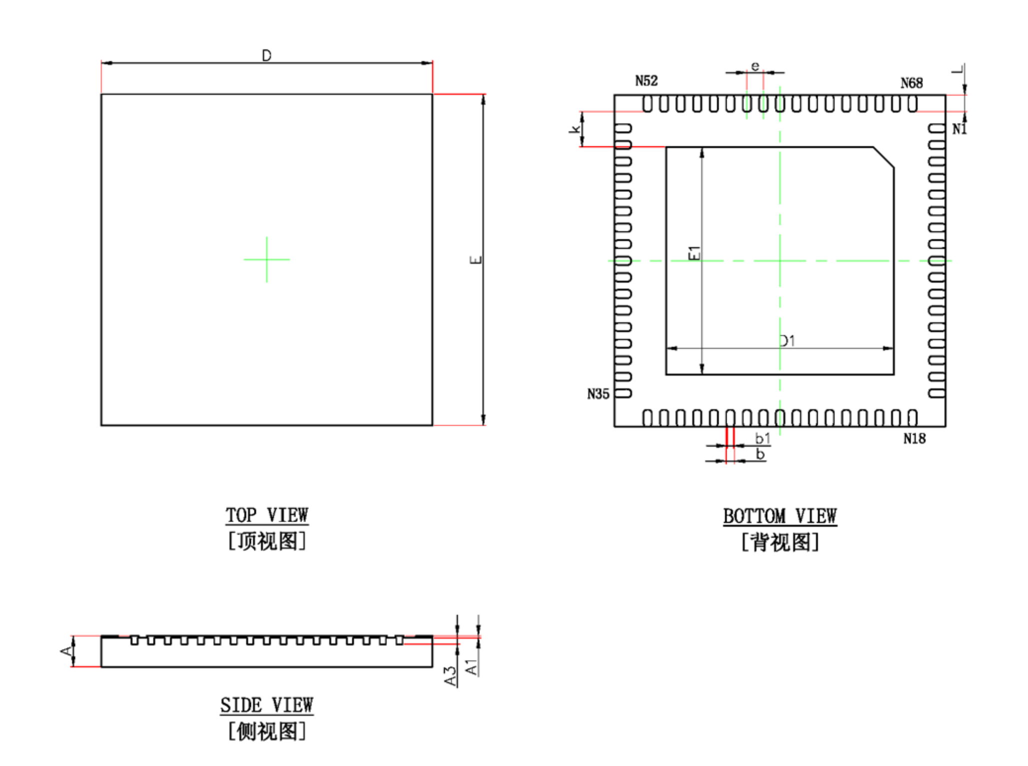
PIN No.	PIN name	ALT0	ALT1 (default)	ALT2	ALT3	备注
1	PA4		PA[4]	PWM4	XTAL32K	
2	PA5		PA[5]	PWM5	CLK_27P12	可配置输出27.12M
3	PA8	SCI0_CLK	PA[8]			复用为IO时必须先打开IC卡电源，且输出信号的高电平为IC卡输出电平
4	PA9	SCI0_RSTN	PA[9]			
5	PA10	SCI0_IO	PA[10]			
6	CVCC					IC卡电源
7	VDD33					
8	POWER_KEY					开关机按键
9	EN_LDO5V					外部LDO使能
10	**VSS**					芯片地
11	PA6	SCI0_DET	PA[6]			
12	PA7		PA[7]	CLK_27P12		可配置输出27.12M
13	PA11		PA[11]			
14	PA12		PA[12]		UART0_CTS	
15	PA13		PA[13]	CLK_27P12	UART0_RTS	可配置输出27.12M
16	PA14	SCI0_VCCEN	PA[14]	UART2_RX	UART0_RX	
17	PA15		PA[15]	UART2_TX	UART0_TX	
18	PA0	UART0_RX/IrDA_IN	PA[0]	PWM0		串口下载管脚
19	PA1	UART0_TX/IrDA_OUT	PA[1]	PWM1		
20	PA2	UART0_CTS	PA[2]	PWM2		
21	PA3	UART0_RTS	PA[3]	PWM3		
22	PB0	I2C0_SCL	PB[0]	PWM0	XTAL32K	
23	PB1	I2C0_SDA	PB[1]	PWM1	CLK_27P12	
24	PD4	UART1_RX/IrDA_IN	PD[4]			
25	PD5	UART1_TX/IrDA_OUT	PD[5]			
26	PB2	SPI0_SCK	PB[2]	PWM2	SPI3_CLK	

27	PB3	SPI0_CSN0	PB[3]	PWM3	SPI3_CSN0	
28	PB4	SPI0_MOSI	PB[4]	PWM4	SPI3_MOSI	
29	PB5	SPI0_MISO	PB[5]	PWM5	SPI3_MISO	
30	IO_VCC					PB2~PB5电源输入
31	PB12	SPI2_CLK	PB[12]	SPI3_CLK	UART1_RX/IrDA_IN	
32	PB13	SPI2_CSN	PB[13]	SPI3_CSN0	UART1_TX/IrDA_OUT	
33	PB14	SPI2_MOSI	PB[14]	SPI3_MOSI	UART1_CTS	
34	PB15	SPI2_MISO	PB[15]	SPI3_MISO	UART1_RTS	
35	PB10		PB[10]			
36	PB11		PB[11]			
37	PC6		PC[6]		CLK_27P12	
38	PC7		PC[7]			
39	PC8	UART2_TX/IrDA_OUT	PC[8]		SPI1_MISO	
40	VDD12					
41	RST_N					
42	PC9	UART2_CTS	PC[9]		SPI1_MOSI	
43	PC10	UART2_RX/IrDA_IN	PC[10]		SPI1_SCK	
44	PB6	SPI1_SCK	PB[6]	SPI3_SCK		
45	PB7	SPI1_CSN0	PB[7]	SPI3_CSN0		
46	PB8	SPI1_MOSI	PB[8]	SPI3_MOSI		
47	PB9	SPI1_MISO	PB[9]	SPI3_MISO		
48	PC15	SPI1_MISO	PC[15]	SPI3_MISO	UART2_TX/IrDA_OUT	
49	PC14	SPI1_MOSI	PC[14]	SPI3_MOSI	UART2_RTS	
50	PC13	SPI1_CLK	PC[13]	SPI3_CLK	UART2_RX/IrDA_IN	
51	PC12	SPI1_CSN	PC[12]	SPI3_CSN	UART2_CTS	
52	PC11	UART2_RTS	PC[11]		SPI1_CSN0	
53	CHARGE_VPROG					CHARGE充电电流控制管脚
54	CHARGE_VCC					CHARGE电源输入
55	CHARGE_VBAT					CHARGE电源输出，接电池
56	PD6	UART1_CTS	PD[6]	MSR_IN1	I2C0_SCL	磁条卡 T1+
57	PD7	UART1_RTS	PD[7]	MSR_IN2	I2C0_SDA	磁条卡 T1-
58	PD8	SPI1_SCK	PD[8]	MSR_IN3		磁条卡 T2+
59	PD9	SPI1_CSN	PD[9]	MSR_IN4		磁条卡 T2-
60	PD10	SPI1_MOSI	PD[10]	MSR_IN5		磁条卡 T3+
61	PD11	SPI1_MISO	PD[11]	MSR_IN6		磁条卡 T3-
62	PC5		PC[5]	ADC_IN5		
63	PC4	TCK	PC[4]	ADC_IN4		
64	PC3	TMS	PC[3]	ADC_IN3	UART1_RTS	
65	PC2	TDO	PC[2]	ADC_IN2	UART1_CTS	
66	PC1	TDI	PC[1]	ADC_IN1/DAC	UART1_TX/IrDA_OUT	

67	PC0	TRST	PC[0]	ADC_IN0	UART1_RX/IrDA_IN	
68	VDD33					IO电源
69	PD0	DRV_VBUS	PD[0]			
70	**VSS**					芯片地
71	DN					USB DN
72	DP					USB DP
73	VBUS					USB VBUS
74	ID					USB ID
75	XI12M					XTAL 12MHz Input
76	XO12M					XTAL12MHz Output
77	VDD33					
78	EXTS7					外部Tamper7
79	EXTS5					外部Tamper5
80	XO32					XTAL 32KHz Output
81	XI32					XTAL 32KHz Input
82	EXTS6					外部Tamper6
83	EXTS4					外部Tamper4
84	EXTS3					外部Tamper3
85	EXTS1					外部Tamper1
86	EXTS2					外部Tamper2
87	EXTS0					外部Tamper0
88	VBAT33					电池供电电源

2.3 封装信息

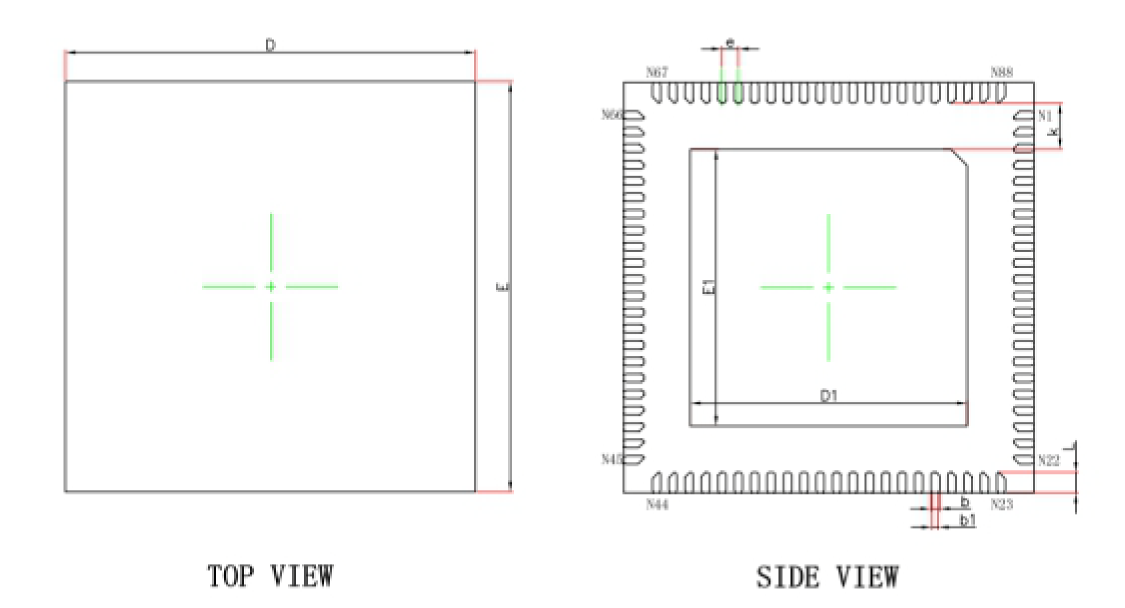
2.3.1 QFN68

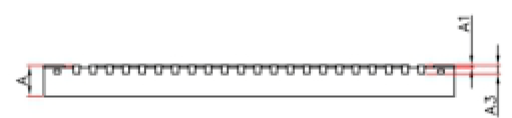


Symbol	Dimensions In Millimeters		Dimensions In Inches	
	Min.	Max.	Min.	Max.
A	0.700	0.800	0.028	0.031
A1	0.000	0.050	0.000	0.002
A3	0.203REF.		0.008REF.	
D	7.900	8.100	0.311	0.319
E	7.900	8.100	0.311	0.319
D1	5.400	5.600	0.213	0.220
E1	5.400	5.600	0.213	0.220
k	0.200MIN.		0.008MIN.	
b	0.150	0.250	0.006	0.010
b1	0.100	0.200	0.004	0.008
e	0.400TYP.		0.016TYP.	
L	0.324	0.476	0.013	0.019

图 2-1 MH1902T QFN68 8mm×8mm封装尺寸

2.3.2 QFN88





Symbol	Dimensions In Millimeters		Dimensions In Inches	
	MIN.	MAX.	MIN.	MAX.
A	0.700	0.800	0.028	0.031
A1	0.000	0.050	0.000	0.002
A3	0.203REF.		0.008REF.	
D	9.900	10.100	0.390	0.398
E	9.900	10.100	0.390	0.398
D1	6.650	6.850	0.262	0.270
E1	6.650	6.850	0.262	0.270
k	1.125REF.		0.044REF.	
b	0.150	0.250	0.006	0.010
b1	0.150REF.		0.006REF.	
e	0.400BSC.		0.016BSC.	
L	0.400	0.600	0.016	0.024

图 2-2 MH1902T QFN88 10mm*10mm封装尺寸

3 系统控制（SYSCTRL）

3.1 软复位

系统提供软件复位操作，对软件复位寄存器（SOFT_RST1或者SOFT_RST2）对应操作位写“1”实现相应模块复位操作，写“1”后由硬件清“0”。部分模块复位需要提前清除保护锁寄存器（LOCK_R）后在进行复位操作，详见寄存器描述。

3.2 时钟

芯片支持内部12MHz振荡器和外置12MHz晶体，内部集成的12MHz OSC常温下精度为±1%，经过PLL倍频后为系统提供输入，倍频后的PLL时钟频率可通过软件进行配置，其频率可配为144MHz/120MHz/108MHz/72MHz/60MHz/54MHz。PLL时钟经过后clk division_0分频逻辑单元后作为FCLK、HCLK时钟，再经过clk division_1分频逻辑单元后作为PCLK时钟。

芯片上电复位后，默认基于内部12MHz的振荡器工作，若需要使用外部12MHz晶体时，需要软件切换到外部12MHz工作。

芯片的整个安全区基于内部32KHz工作，内部集成的32KHz OSC常温下精度±1KHz；RTC默认基于内部32KHz工作，可软件切换使用外部32KHz工作。支持内部或外部32KHz输出。

MH1902T时钟树如下图：

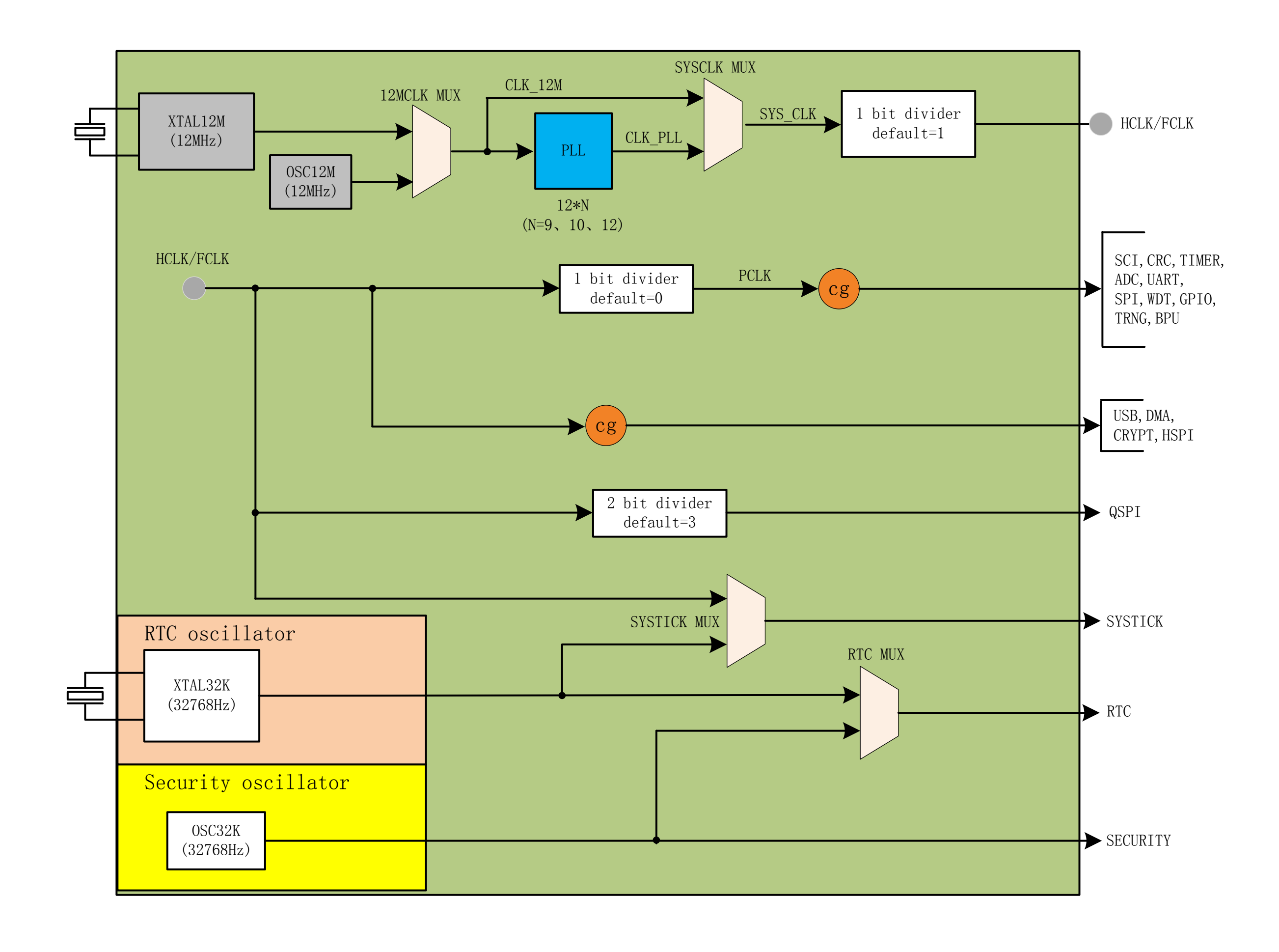


图 3-1 MH1902T时钟树

3.2.1 外部时钟源

芯片外部系统时钟要求为12MHz，外部实时时钟要求为32KHz。

3.2.2 PLL时钟

外部时钟经PLL倍频后得到PLL时钟，可通过寄存器FREQ_SEL配置产生6种不同的PLL时钟:

- 144MHz
- 120MHz
- 108MHz
- 72MHz
- 60MHz
- 64MHz

3.2.3 FCLK时钟

FCLK 时钟由 PLL 倍频提供。

3.2.4 HCLK/FCLK时钟

PLL时钟通过时钟分频单元分频后作为HCLK和FCLK时钟，因此HCLK和FCLK时钟频率始终相等。

通过寄存器对PLL时钟进行分频：

- HCLK/FCLK= PLL 时钟
- HCLK/FCLK= PLL 时钟 / 2
- HCLK/FCLK= PLL 时钟 / 4

3.2.5 PCLK时钟

HCLK时钟通过时钟分频单元分频后作为PCLK时钟。

通过寄存器对PLL时钟进行分频：

- PCLK = HCLK / 2
- PCLK = HCLK / 4

3.2.6 外设时钟管理

系统提供时钟门控控制寄存器（CG_CTRL）管理外设时钟。用户可以通过该寄存器打开和关闭对应外设时钟。外设时钟关闭后外设将不在运行，通过该寄存器可以更灵活的控制系统功耗。

3.3 低功耗控制

系统或电源复位后，安全CPU处于运行状态。当CPU不需要继续运行时，可以利用多种低功耗模式来节省功耗，例如等待某个外部事件的产生。

可以通过下几种方式降低功耗：

- 降低系统、AHB 及 APB 总线时钟（FREQ_SEL）
- 关闭 AHB 和 APB 总线上不使用的外设时钟（CG_CTRL）
- 睡眠模式（CPU 内核停止，所有外设仍在运行）
- 深度睡眠模式（CPU 内核和外设均停止运行）

功耗状态转换图：

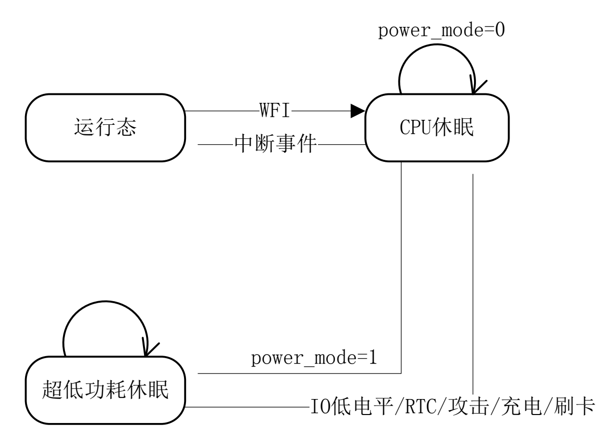


图 3-1 功耗转换示意图

在执行WFI指令之前必须先配置FREQ_SEL.power_mode寄存器，选择CPU休眠或超低功耗休眠，当power_mode = 0时则执行CPU休眠，任意中断都能唤醒CPU；当power_mode = 1时则执行超低功耗休眠，先关CPU时钟进入CPU休眠模式，再关闭PLL进入超低功耗休眠模式，在超低功耗状态下只有IO低电平、RTC、攻击、充电或刷卡能将芯片从超低功耗休眠状态切换到CPU休眠状态，产生中断后唤醒CPU。

超低功耗休眠的唤醒源由WKUP_TYPE_EN、WKUP_PL_EN和WKUP_PH_EN 3个寄存器进行控制。

3.4 外设控制

系统控制区域包含1个外设控制寄存器（PHER_CTRL），该寄存器实现对部分外设的控制。

控制内容如下：

- SCI0 VCCEN信号有效电平选择
- SCI0卡检测信号有效电平选择
- SPI0工作模式选择
- HSPI工作模式选择

3.5 寄存器描述

3.5.1 地址映射表

SYSCTRL基地址表

地址范围	基地址	外设	总线
0x4001_F000-0x4001_FFFF	0x4001_F000	SYSCTRL	APB0

表 3-1 SYSCTRL寄存器表

偏移地址	寄存器名称	宽度（bit）	复位值
0x00	FREQ_SEL	32	0xFFF80600
0x04	CG_CTRL1	32	0x04100000
0x08	CG_CTRL2	32	0x00000000
0x0C	SOFT_RST1	32	0x00000000
0x10	SOFT_RST2	32	0x00000000
0x14	LOCK_R	32	0xF0000000
0x18	PHER_CTRL	32	0x00108000
0x1C-0x2B	SYS_RSVD0	32	0x00000000
0x2C	HCLK_1MS_VAL	32	0x00011940

0x30	PCLK_1MS_VAL	32	0x00008CA0
0x34	ANA_CTRL	32	0x000204B3
0x38	DMA_CHAN	32	0x0A0B0203
0x3C	SCI_GLF	32	0x20060000
0x40	SW_RSV	32	0x00000000
0x44	SW_RSV1	32	0x00000000
0x48	CARD_RSVD	32	0x01040210
0x4C	ANA_CTRL1	32	0x00000208
0x50-0xFF	SYS_RSVD	32	0x00000000
0100	MSR_CR1	32	0x07F88800
0104	MSR_CR2	32	0x0000000C
0108	保留	32	0x204921AE
101C	保留	32	0x40000000
0110	保留	32	0x00000000
0114-03E8	SYS_RSVD2	32	0x00000000

3.5.2 时钟频率选择寄存器（FREQ_SEL）

偏移地址：0x0000　　复位值：0xFFF80600

31	30	29	28	27	26	25	24
预留							
ro							

23	22	21	20	19	18	17	16
预留		rom_pd_en	otp_pd_en	usb_pd_en	power_mode[2:0]		
ro		rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8
预留					sys_ck_src	ck12m_src	xtal_sel
ro					rw	rw	rw

7	6	5	4	3	2	1	0
xtal_sel				hclk_div_en	hclk_div_val	保留	pclk_div_val
rw				rw	rw	ro	rw

Bit	Name	W/R	Description
31:22	预留	-	预留
21	rom_pd_en	W/R	deep sleep 模式下 rom 关电使能： 0：不掉电； 1：掉电；
20	otp_pd_en	W/R	deep sleep 模式下 OTP 关电使能： 0：不掉电； 1：掉电；
19	usb_pd_en	W/R	deep sleep 模式下 USB 关电使能： 0：不掉电； 1：掉电；
18:16	power_mode[2:0]	W/R	低功耗模式选择： 3'h0:CPU 休眠； 3'h1:DEEP SLEEP（时钟关闭，ROM、USB、OTP 可选断电）； 3'h2:掉电模式（仅 IO 低电平、RTC、SENSOR 攻击

			可唤醒）； 3'h3~3'h7:预留； 注：进入低功耗模式前，先配置此寄存器选择功耗模式，然后通过处理器的 WFI 指令进入。
15:11	预留	-	预留
10	sys_ck_src	W/R	系统主时钟来源选择： 0 ： 系统时钟来源于 PLL 之前； 1 ： 系统时钟来源于 PLL 之后
9	ck12m_src	W/R	12MHz 时钟来源选择： 0 ： 片外 XTAL； 1 ： 片内 OSC
8:4	xtal_sel	W/R	当 clk_sys_src 为 1 时，系统时钟频率选择： 5'b00100: 144MHz; 5'b00101: 120MHz; 5'b0011x: 108MHz; 5'b00000: 72MHz; 5'b00001: 60MHz; 5'b0001x: 54MHz;
3	hclk_div_en	W/R	HCLK 分频使能
2	hclk_div_val	W/R	HCLK 分频值 0：HCLK 在系统时钟的基础上 2 分频； 1：HCLK 在系统时钟的基础上 4 分频
1	预留	-	预留
0	pclk_div_val	W/R	PCLK 分频值（基于 HCLK）： 0：PCLK 在 HCLK 的基础上 2 分频； 1：PCLK 在 HCLK 的基础上 4 分频

3.5.3 时钟门控控制寄存器1（CG_CTRL1）

偏移地址：0x0004 复位值：0x04100000

31	30	29	28	27	26	25	24
trng_cg_en	adc_cg_en	crc_cg_en	预留		bpu_cg_en	预留	
rw	rw	rw	ro		rw	ro	

23	22	21	20	19	18	17	16
预留		timer_cg_en	gpio_cg_en	预留			
ro		rw	rw	ro			

15	14	13	12	11	10	9	8
预留	sci0_cg_en	预留			spi2_cg_en	spi1_cg_en	spi0_cg_en
ro	rw	ro			rw	rw	rw

7	6	5	4	3	2	1	0
预留					uart2_cg_en	uart1_cg_en	uart0_cg_en
ro							

Bit	Name	W/R	Description
31	trng_cg_en	W/R	随机数模块门控时钟使能： 0: 不使能； 1：使能
30	adc_cg_en	W/R	ADC 门控时钟使能： 0: 不使能； 1：使能
29	crc_cg_en	W/R	CRC 门控时钟使能： 0: 不使能； 1：使能
28:27	预留	-	预留

26	bpu_cg_en	W/R	电池供电模块门控时钟使能： 0: 不使能； 1：使能
25:22	预留	-	预留
21	timer_cg_en	W/R	Timer 门控时钟使能： 0: 不使能； 1：使能
20	gpio_cg_en	W/R	GPIO 模块门控时钟使能： 0: 不使能； 1：使能
19:15	预留	-	预留
14	sci0_cg_en	W/R	SCI0 门控时钟使能： 0: 不使能； 1：使能
13:11	预留	-	预留
10	spi2_cg_en	W/R	SPI2 门控时钟使能： 0: 不使能； 1：使能
9	spi1_cg_en	W/R	SPI1 门控时钟使能： 0: 不使能； 1：使能
8	spi0_cg_en	W/R	SPI0 门控时钟使能： 0: 不使能； 1：使能
7:3	预留	-	预留
2	uart2_cg_en	W/R	UART2 门控时钟使能： 0: 不使能； 1：使能
1	uart1_cg_en	W/R	UART1 门控时钟使能： 0: 不使能； 1：使能
0	uart0_cg_en	W/R	UART0 门控时钟使能： 0: 不使能； 1：使能

3.5.4 时钟门控控制寄存器2（CG_CTRL2）

偏移地址：0x0008 复位值：0x00000000

31	30	29	28	27	26	25	24
预留		dma_cg_en	usb_cg_en	预留	asymc_crypt_cg_en	预留	
ro		rw	rw	ro	rw	ro	

23	22	21	20	19	18	17	16
预留							
ro							

15	14	13	12	11	10	9	8
预留							
ro							

7	6	5	4	3	2	1	0
预留					hspi_cg_en	预留	symc_crypt_cg_en
ro					rw	ro	rw

Bit	Name	W/R	Description
31:30	预留	--	预留
29	dma_cg_en	RW	DMA 门控时钟使能： 0: 不使能； 1：使能
28	usb_cg_en	RW	USB PHY 及 USB controler 门控时钟使能： 0: 不使能； 1：使能
27	预留	--	预留
26	asymc_crypt_cg_en	RW	非对称算法加速引擎门控时钟使能：

			0：不使能；　1：使能
25:3	预留	--	预留
2	hspi_cg_en	RW	高速 SPI 门控时钟使能 0：不使能；　1：使能
1	预留	--	预留
0	symc_crypt_cg_en	RW	对称算法加速引擎门控时钟使能： 0：不使能；　1：使能

3.5.5　软件复位寄存器1（SOFT_RST1）

偏移地址：0x000C　　　　　　　　复位值：0x00000000

31	30	29	28	27	26	25	24
srst_trng	srst_adc	srst_crc	预留				
wc	wc	wc	ro				
23	22	21	20	19	18	17	16
预留		srst_timer	srst_gpio	预留			
ro		wc	wc	ro			
15	14	13	12	11	10	9	8
预留	srst_sci0	预留			srst_spi2	srst_spi1	srst_spi0
ro	wc	ro			wc	wc	wc
7	6	5	4	3	2	1	0
预留					srst_uart2	srst_uart1	srst_uart0
ro					wc	wc	wc

对于所有的软复位信号，对对应的bit写1执行软复位操作（写1后硬件自动恢复为0，不需要软件清0）

Bit	Name	W/R	Description
31	srst_trng	WC	随机数软复位信号
30	srst_adc	WC	ADC 模块软复位信号
29	srst_crc	WC	CRC 模块软复位信号
28:22	预留	--	预留
21	srst_timer	WC	Timer0 软复位信号
20	srst_gpio	WC	GPIO 模块软复位信号
19:15	预留	--	预留
14	srst_sci0	WC	SCI0 软复位信号
13:11	预留	--	预留
10	srst_spi2	WC	SPI2 软复位信号
9	srst_spi1	WC	SPI1 软复位信号
8	srst_spi0	WC	SPI0 软复位信号
7:3	预留	--	预留
2	srst_uart2	WC	UART2 软复位信号
1	srst_uart1	WC	UART1 软复位信号
0	srst_uart0	WC	UART0 软复位信号

3.5.6　软件复位寄存器2（SOFT_RST2）

偏移地址：0x0010　　　　　　　　复位值：0x00000000

31	30	29	28	27	26	25	24
srst_glb	srst_cm3	srst_dma	srst_usb	srst_cache	srst_asymc_crypt	预留	
wc	wc	wc	wc	wc	wc	ro	

23	22	21	20	19	18	17	16
预留							
ro							

15	14	13	12	11	10	9	8
预留							
ro							

7	6	5	4	3	2	1	0
预留					srst_hspi	srst_otp	srst_symc_crypt
ro					wc	wc	wc

对于所有的软复位信号，对对应的bit写1执行软复位操作（写1后硬件自动恢复为0，不需要软件清0）

Bit	Name	W/R	Description
31	srst_glb	WC	芯片数字部分全局软复位信号，结合 lock 信号使用
30	srst_cm3	WC	CPU 内核软复位信号，结合 lock 信号使用
29	srst_dma	WC	DMA 软复位信号，结合 lock 信号使用
28	srst_usb	WC	USB 接口软复位信号，结合 lock 信号使用
27	srst_cache	WC	Cache软复位
26	srst_asymc_crypt	WC	非对称算法软复位
25:3	预留	--	预留
2	srst_hspi	WC	高速SPI接口软复位信号
1	srst_otp	WC	OTP软复位信号
0	srst_symc_crypt	WC	对称算法软复位

3.5.7 保护锁寄存器（LOCK_R）

偏移地址：0x0014　　　　复位值：0xF0000000

31	30	29	28	27	26	25	24
srst_glb_lock	srst_cm3_lock	srst_dma_lock	srst_usb_lock	预留			
rw	rw	rw	rw	ro			

23	22	21	20	19	18	17	16
预留							
ro							

15	14	13	12	11	10	9	8
预留							
ro							

7	6	5	4	3	2	1	0
预留							
ro							

Bit	Name	W/R	Description
31	srst_glb_lock	W/R	芯片全局软复位保护控制位： 0 ：允许软件执行全局软复位； 1 ：不允许软件执行全局软复位；
30	srst_cm3_lock	W/R	CPU 内核软复位保护控制位： 0 ：允许软件执行 CPU 内核软复位； 1 ：不允许软件执行 CPU 内核软复位；
29	srst_dma_lock	W/R	DMA 软复位保护控制位： 0 ：允许软件执行 DMA 软复位； 1 ：不允许软件执行 DMA 软复位；

28	srst_usb_lock	W/R	USB 软复位保护控制位： 0 ：允许软件执行 USB 软复位； 1 ：不允许软件执行 USB 软复位；
27:0	预留	--	预留

3.5.8 外设控制寄存器（PHER_CTRL）

偏移地址：0x0018 复位值：0x00108000

31	30	29	28	27	26	25	24
预留				hspi_slv_sel	预留		spi0_slv_sel
ro				rw	ro		rw

23	22	21	20	19	18	17	16
预留			sci0_vccen_inv	预留			sci0_cdet_inv
ro			rw	ro			rw

15	14	13	12	11	10	9	8
sci_glf_vccen	sci_cn_glf_en	预留					
rw	rw	ro					

7	6	5	4	3	2	1	0
预留							
ro							

Bit	Name	W/R	Description
31:28	预留	--	预留
27	hspi_slv_sel	RW	HSPI 工作模式选择 0：Master 模式；1：Slave 模式
26:25	预留	--	预留
24	spi0_slv_sel	RW	SPI0 工作模式选择 0：Master 模式；1：Slave 模式
23:21	预留	--	预留
20	sci0_vccen_inv	RW	SCI0 VCCEN信号有效电平选择 0: 高有效；1: 低有效
19:17	预留	--	预留
16	sci0_cdet_inv	RW	SCI0卡检测信号有效电平选择 0: 高有效；1: 低有效
15	sci_glf_vccen	RW	1:SCI Card Normal毛刺滤除计数使能受VCCEN硬件控制，VCCEN无效时，毛刺计数清0并停止计数； 0:SCI Card Normal毛刺滤除计数使能不受VCCEN硬件控制
14	sci_cn_glf_en	RW	1:SCI Card Normal毛刺滤除计数使能 0:SCI Card Normal毛刺滤除计数不使能，毛刺计数器同时清0
13:0	预留	--	预留

3.5.9 HCLK 1ms对应的cycle（HCLK_1MS_VAL）

偏移地址：0x002C 复位值：0x00011940

31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16

预留															
ro															ro

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
val_1ms_hclk															
ro															

Bit	Name	W/R	Description
31:17	预留	--	预留
16:0	val_1ms_pclk	RO	基于 HCLK 计时 1ms 所需要的 cycle 值

3.5.10 PCLK 1ms对应的cycle（PCLK_1MS_VAL）

偏移地址：0x0030　　　　复位值：0x00008CA0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
预留															
ro															ro

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
val_1ms_pclk															
ro															

Bit	Name	W/R	Description
31:17	预留	--	预留
16:0	val_1ms_pclk	RO	基于 PCLK 计时 1ms 所需要的 cycle 值

3.5.11 DMA通道选择（DMA_CHAN）

偏移地址：0x0038　　　　复位值：0x0A0B0203

31	30	29	28	27	26	25	24
预留			dma_ch3_if				
ro							

23	22	21	20	19	18	17	16
预留			dma_ch2_if				
ro							

15	14	13	12	11	10	9	8
预留			dma_ch1_if				
ro			rw				

7	6	5	4	3	2	1	0
预留			dma_ch0_if				
ro			rw				

Bit	Name	W/R	Description
31:29	预留	--	预留
28:24	dma_ch3_if	W/R	DMA hardware handshaking interface select for channel 3: 5'h0: HSPI_TX; 5'h1: HSPI_RX; 5'h2: UART0 TX; 5'h3: UART0 RX; 5'h4: UART1 TX; 5'h5: UART1 RX; 5'h6: UART2 TX; 5'h7: UART2 RX;

			5'ha: SPI0 TX; 5'hb: SPI0 RX; 5'hc: SPI1 TX; 5'hd: SPI1 RX; 5'he: SPI2 TX; 5'hf: SPI2 RX; 5'h1a: QSPI;
23:21	预留	--	预留
20:16	dma_ch2_if	W/R	DMA hardware handshaking interface select for channel 2: 5'h0: HSPI_TX; 5'h1: HSPI_RX; 5'h2: UART0 TX; 5'h3: UART0 RX; 5'h4: UART1 TX; 5'h5: UART1 RX; 5'h6: UART2 TX; 5'h7: UART2 RX; 5'ha: SPI0 TX; 5'hb: SPI0 RX; 5'hc: SPI1 TX; 5'hd: SPI1 RX; 5'he: SPI2 TX; 5'hf: SPI2 RX; 5'h1a: QSPI;
15:13	预留	--	预留
12:8	dma_ch1_if	W/R	DMA hardware handshaking interface select for channel 1: 5'h0: HSPI_TX; 5'h1: HSPI_RX; 5'h2: UART0 TX; 5'h3: UART0 RX; 5'h4: UART1 TX; 5'h5: UART1 RX; 5'h6: UART2 TX; 5'h7: UART2 RX; 5'ha: SPI0 TX; 5'hb: SPI0 RX; 5'hc: SPI1 TX; 5'hd: SPI1 RX; 5'he: SPI2 TX; 5'hf: SPI2 RX; 5'h1a: QSPI;
7:5	预留	--	预留
4:0	dma_ch0_if	W/R	DMA hardware handshaking interface select for channel 0: 5'h0: HSPI_TX; 5'h1: HSPI_RX; 5'h2: UART0 TX; 5'h3: UART0 RX; 5'h4: UART1 TX; 5'h5: UART1 RX; 5'h6: UART2 TX; 5'h7: UART2 RX; 5'ha: SPI0 TX; 5'hb: SPI0 RX; 5'hc: SPI1 TX; 5'hd: SPI1 RX;

			5'he: SPI2 TX; 5'hf: SPI2 RX; 5'h1a: QSPI;

3.5.12 SCI毛刺滤除配置（SCI_GLF）

偏移地址：0x003C 复位值：0x20060000

31	30	29	28	27	26	25	24
card_normal_glf_bypass	card_normal_bypass	vsel		预留			
rw	rw	rw		ro			

23	22	21	20	19	18	17	16
预留				card_normal_glf			
ro				rw			

15	14	13	12	11	10	9	8
card_normal_glf							
rw							

7	6	5	4	3	2	1	0
card_normal_glf							
rw							

Bit	Name	W/R	Description
31	card_normal_glf_bypass	RW	card_normal 信号（0:过流，1:正常）毛刺滤除 bypass 1：关闭 card_normal 的毛刺滤除 0：打开毛刺滤除
30	card_normal_bypass	RW	card_noraml 不参与 card detected 信号的生成。 card_noraml 由模拟反馈，指示 7816 卡电压正常
29:28	vsel	RW	卡供电电压选择 00:保留 01:保留 10:3V 11:1.8V
27:20	预留	--	预留
19:0	card_normal_glf	W/R	card_normal 毛刺滤除计数最大值，计数器在计数在最大值之前出现的毛刺将被滤除

3.5.13 软件预留寄存器（CARD_RSVD）

偏移地址：0x0048 复位值：0x0100021E

31	30	29	28	27	26	25	24
保留					chg_state		chg_intp_en
ro					ro		rw

23	22	21	20	19	18	17	16
chg_intp	保留						
rw	ro						

15	14	13	12	11	10	9	8
保留							
ro							

7	6	5	4	3	2	1	0
保留							

ro

Bit	Name	W/R	Description
31:27	保留	--	保留
26:25	chg_state	RO	充电状态： 11:充满 10:恒流充电 01:涓流充电 00:欠压
24	chg_intp_en	RW	1:当chg_state不为0时上报中断 0:不上报chg_state中断，但可从chg_intp中查到状态位
23	chg_intp	RW	chg中断状态位，硬件置1，软件写0清0
22:0	保留	--	保留

3.5.14 模拟控制寄存器（ANA_CTRL1）

偏移地址：0x004C 复位值：0x00000208

31	30	29	28	27	26	25	24
保留							
ro							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
保留							
ro							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
保留						pd_ldo25_sel	pd_ldo25
ro						rw	rw
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
保留							
ro							

Bit	Name	W/R	Description
31:10	保留	--	保留
9	pd_ldo25_sel	RW	1: PD LDO25由pd_ldo25控制
8	pd_ldo25	RW	0:Power Up LDO25 1:Power Down LDO25
7:0	保留	--	保留

3.5.15 MSR控制寄存器1（MSR_CR1）

偏移地址：0x0100 复位值：0x07F88800

31	30	29	28	27	26	25	24
保留				pd_msr	保留		
ro				rw	ro		
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
保留							
ro							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
保留							
ro							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**

保留

ro

Bit	Name	W/R	Description
31:28	保留	--	保留
27	pd_msr	RW	MSR关电使能： 0 ： 不关电； 1 ： 关电且将电源拉到地
26:0	保留	--	保留

4 通用输入输出（GPIO）

4.1 GPIO功能描述

芯片每组GPIO包含4个32位寄存器和1组5bit寄存器：

- 数据寄存器（Px_IODR）
- 置位/复位寄存器（Px_BSRR）
- 方向寄存器（Px_OEN）
- 上拉电阻使能寄存器（Px_PUE）。

Px_IODR高16位作为输入寄存器（Px_IDR）（只读），低16位作为输出寄存器（Px_ODR）（读写）

Px_BSRR高16位作为reset寄存器，低16位作为set寄存器

GPIO端口的每个引脚可以配置为多种工作方式：

- 输入模式（输入高阻、输入上拉）
- 开漏输出
- 推挽输出

GPIO工作模式配置图：

工作模式	Px_IODR	Px_BSRR	Px_OEN	Px_PUE
输入高阻	data_r	不使用	1	0
输入上拉	data_r	不使用	1	1
开漏输出	data_r	不使用	data_w	0
推挽输出	data_rw	data_w	0	0

注：
data_r：数据读操作
data_w：数据写操作
data_rw：数据读写操作

4.1.1 通用I/O（GPIO）

作为输出配置时，写到输出数据寄存器上的值将输出到对于I/O上。输入数据寄存器显示APB上捕捉I/O上的数据。

所有GPIO引脚上都有一个内部上拉，可以通过上拉使能寄存器（Px_PUE）控制是否有效。

所有通用IO复位后默认状态为上拉输入，上拉电阻46K。

4.1.2 专用I/O（GPIO）

PA8、PA9、PA10这3个I/O是7816模块所使用的I/O，所以设计上是单独供电的，若用作普通I/O需打开7816模块的电源才能正常工作。PA10需打开内部上拉电阻才能维持高电平，需关闭内部上拉电阻才能维持低电平，可以通过上拉使能寄存器（Px_PUE）控制是否打开内部上拉电阻。

注：PA6是普通I/O，不由7816模块供电。

4.1.3 单独的位设置或清除

通过置位/复位寄存器（Px_BSRR）可以实现对单独I/O的置位/复位操作。

4.1.4 外部中断

通过配置 GPIO 使其对引脚上状态变化对 CPU 产生中断请求。
GPIO 外部中断响应类型：

- 上升沿中断
- 下降沿中断
- 双边沿中断

4.1.5 外部唤醒事件

芯片所有GPIO管脚均支持超低功耗唤醒，GPIO仅支持低电平唤醒，可通过WKUP_PL_EN和WKUP_PH_EN寄存器配置IO唤醒源。

4.1.6 I/O功能复用

外设与I/O复用可通过复用控制寄存器（Px_ALT）进行配置。

- 根据需要进行 I/O 复用
- 复用为功能外设后，无需配置 I/O 工作模式（输入高阻、输入上拉、开漏输出、推挽输出），复用配置完成后系统会自动进入对应 I/O 工作模式。

4.2 GPIO寄存器

4.2.1 地址映射表

GPIO基地址表

地址范围	基地址	外设	总线
0x4001_D000-0x4001_DFFF	0x4001_D000	GPIO	APB0

表 4-1 GPIO寄存器表

偏移地址	寄存器名称	宽度（bit）	复位值	注释
0x00	PA_IODR	32	0xFFFF0000	GPIOA
0x04	PA_BSRR	32	0x00000000	
0x08	PA_OEN	32	0x0000FFFF	
0x0C	PA_PUE	32	0x0000FFFF	
0x10	PB_IODR	32	0xFFFF0000	GPIOB
0x14	PB_BSRR	32	0x00000000	
0x18	PB_OEN	32	0x0000FFFF	
0x1C	PB_PUE	32	0x0000FFFF	
0x20	PC_IODR	32	0xFFFF0000	GPIOC
0x24	PC_BSRR	32	0x00000000	
0x28	PC_OEN	32	0x0000FFFF	
0x2C	PC_PUE	32	0x0000FFFF	
0x30	PD_IODR	32	0xFFFF0000	GPIOD
0x34	PD_BSRR	32	0x00000000	
0x38	PD_OEN	32	0x0000FFFF	
0x3C	PD_PUE	32	0x0000FFFF	
0x40 - 0x118	RSVD0	32	0x00000000	
0x11C	INTP3_STA	32	0x00000000	EXTI
0x120	INTP2_STA	32	0x00000000	
0x124	INTP1_STA	32	0x00000000	
0x128	INTP0_STA	32	0x00000000	

0x12C - 0x17C	RSVD1	32	0x00000000	
0x180	PA_ALT	32	0x55555555	ALT
0x184	PB_ALT	32	0x55555555	
0x188	PC_ALT	32	0x55555555	
0x18C	PD_ALT	32	0x55555555	
0x190 - 0x21C	RSVD2	32	0x00000000	
0x220	WKUP_TYPE_EN	32	0x00007001	WKUP
0x224	WKUP_PL_EN	32	0x00000000	
0x228	WKUP_PH_EN	32	0x00000000	
0x22C-0x7FC	RSVD3	32	0x00000000	
0x800	PA_INTP_TYPE	32	0x00000000	INTP
0x804	PA_INTP_STA	32	0x00000000	
0x808	PB_INTP_TYPE	32	0x00000000	
0x80C	PB_INTP_STA	32	0x00000000	
0x810	PC_INTP_TYPE	32	0x00000000	
0x814	PC_INTP_STA	32	0x00000000	
0x818	PD_INTP_TYPE	32	0x00000000	
0x81C	PD_INTP_STA	32	0x00000000	

4.2.2 数据寄存器（Px_IODR）(x=A..D)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Px_IDR															
ro															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Px_ODR															
rw															

Bit	Name	W/R	Description
31:16	Px_IDR	RO	GPIO 输入数据寄存器
15:0	Px_ODR	W/R	GPIO 输出数据寄存器

4.2.3 置位/复位寄存器（Px_BSRR）(x=A..D)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Px_BR															
wo															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Px_BS															
wo															

Bit	Name	W/R	Description
31:16	Px_BR	WO	GPIO 对应 bit 复位寄存器，当向对应 bit 写 1 时，Px_ODR 对应的 bit 位被清零
15:0	Px_BS	WO	GPIO 对应 bit 置位寄存器，当向对应 bit 写 1 时，Px_ODR 对应的 bit 位被置 1

4.2.4 方向寄存器（Px_OEN）(x=A..D)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
预留															
ro															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Px_OEN															
rw															

Bit	Name	W/R	Description
31:16	预留	--	预留
15:0	Px_OEN	W/R	GPIO 对应 bit 输出使能寄存器： 1 ： 输入； 0 ： 输出。

4.2.5 上拉使能寄存器（Px_PUE）(x=A..D)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
预留															
ro															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Px_PUE															
rw															

Bit	Name	W/R	Description
31:16	预留	--	预留
15:0	Px_PUE	W/R	GPIO 对应 bit 上拉使能

4.2.6 中断状态寄存器（INTPx_STA）(x=A..D)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
预留															
ro															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
预留															intx_state
ro															rc

Bit	Name	W/R	Description
31:1	预留	-	预留
0	intx_state	RC	中断状态寄存器： 0 ：未产生中断； 1：产生中断

4.2.7 GPIOx复用控制寄存器（Px_ALT）(x=A..D)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
px15_alt		px14_alt		px13_alt		px12_alt		px11_alt		px10_alt		px9_alt		px8_alt	
rw		rw		rw		rw		rw		rw		rw		Rw	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

px7_alt	px6_alt	px5_alt	px4_alt	px3_alt	px2_alt	px1_alt	px0_alt
rw	rw	rw	rw	rw	rw	rw	rw

Bit	Name	W/R	Description
31:30	px15_alt	W/R	Px[15]复用控制寄存器
29:28	px14_alt	W/R	Px[14]复用控制寄存器
27:26	px13_alt	W/R	Px[13]复用控制寄存器
25:24	px12_alt	W/R	Px[12]复用控制寄存器
23:22	px11_alt	W/R	Px[11]复用控制寄存器
21:20	px10_alt	W/R	Px[10]复用控制寄存器
19:18	px9_alt	W/R	Px[9]复用控制寄存器
17:16	px8_alt	W/R	Px[8]复用控制寄存器
15:14	px7_alt	W/R	Px[7]复用控制寄存器
13:12	px6_alt	W/R	Px[6]复用控制寄存器
11:10	px5_alt	W/R	Px[5]复用控制寄存器
9:8	px4_alt	W/R	Px[4]复用控制寄存器
7:6	px3_alt	W/R	Px[3]复用控制寄存器
5:4	px2_alt	W/R	Px[2]复用控制寄存器
3:2	px1_alt	W/R	Px[1]复用控制寄存器
1:0	px0_alt	W/R	Px[0]复用控制寄存器

4.2.8 超低功耗唤醒类型控制寄存器（WKUP_TYPE_EN）

31	30	29	28	27	26	25	24
预留							
ro							

23	22	21	20	19	18	17	16
预留							
ro							

15	14	13	12	11	10	9	8
charge_wk_en	sensor_wk_en	msr_wk_en	rtc_wk_en	预留			
rw	rw	rw	rw	ro			

7	6	5	4	3	2	1	0
预留							gpio_wkup_type
ro							rw

Bit	Name	W/R	Description
31:16	预留	--	预留
15	charge_wk_en	RW	充电模块唤醒使能
14	sensor_wk_en	RW	Sensor 中断唤醒使能
13	msr_wk_en	RW	MSR 中断唤醒使能
12	rtc_wk_en	RW	RTC 中断唤醒使能
11:1	预留	--	预留
0	gpio_wkup_type	RW	唤醒源 GPIO 类型及使能控制位： 0：直接唤醒； 1：过滤 1 个 32K 时钟周期毛刺后唤醒

4.2.9 超低功耗唤醒源使能（WKUP_PL_EN）

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
pb_wk_en															
rw															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
pa_wk_en															
rw															

Bit	Name	W/R	Description
31:16	pb_wk_en	W/R	PB 超低功耗唤醒 pb_wk_en[0]对应 PB[0]，pb_wk_en[1]对应 PB[1]
15:0	pa_wk_en	W/R	PA 超低功耗唤醒 pa_wk_en[0]对应 PA[0]，pa_wk_en[1]对应 PA[1]

4.2.10 超低功耗唤醒源使能1（WKUP_PH_EN）

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
pd_wk_en															
rw															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
pc_wk_en															
rw															

Bit	Name	W/R	Description
31:16	pd_wk_en	W/R	PD 超低功耗唤醒 pd_wk_en[0]对应 PD[0]，pd_wk_en[1]对应 PD[1]
15:0	pc_wk_en	W/R	PC 超低功耗唤醒 pc_wk_en[0]对应 PC[0]，pc_wk_en[1]对应 PC[1]

4.2.11 中断类型控制寄存器（Px_INTP_TYPE）(x=A..D)

31 30	29 28	27 26	25 24	23 22	21 20	19 18	17 16
px15_int_type	px14_int_type	px13_int_type	px12_int_type	px11_int_type	px10_int_type	px9_int_type	px8_int_type
rw	rw	rw	rw	rw	rw	rw	rw

15 14	13 12	11 10	9 8	7 6	5 4	3 2	1 0
px7_int_type	px6_int_type	px5_int_type	px4_int_type	px3_int_type	px2_int_type	px1_int_type	px0_int_type
rw	rw	rw	rw	rw	rw	rw	rw

Bit	Name	W/R	Description
31:30	px15_int_type	W/R	Px[15] 中断类型及使能控制位： 00 ：不产生中断；01 ：上升沿中断； 10 ：下降沿中断；11 ：双沿中断
29:28	px14_int_type	W/R	Px[14] 中断类型及使能控制位： 00 ：不产生中断；01 ：上升沿中断； 10 ：下降沿中断；11 ：双沿中断
27:26	px13_int_type	W/R	Px[13] 中断类型及使能控制位： 00 ：不产生中断；01 ：上升沿中断； 10 ：下降沿中断；11 ：双沿中断

25:24	px12_int_type	W/R	Px[12] 中断类型及使能控制位： 00 ：不产生中断；01 ：上升沿中断； 10 ：下降沿中断；11 ：双沿中断
23:22	px11_int_type	W/R	Px[11] 中断类型及使能控制位： 00 ：不产生中断；01 ：上升沿中断； 10 ：下降沿中断；11 ：双沿中断
21:20	px10_int_type	W/R	Px[10] 中断类型及使能控制位： 00 ：不产生中断；01 ：上升沿中断； 10 ：下降沿中断；11 ：双沿中断
19:18	px9_int_type	W/R	Px[9] 中断类型及使能控制位： 00 ：不产生中断；01 ：上升沿中断； 10 ：下降沿中断；11 ：双沿中断
17:16	px8_int_type	W/R	Px[8] 中断类型及使能控制位： 00 ：不产生中断；01 ：上升沿中断； 10 ：下降沿中断；11 ：双沿中断
15:14	px7_int_type	W/R	Px[7] 中断类型及使能控制位： 00 ：不产生中断；01 ：上升沿中断； 10 ：下降沿中断；11 ：双沿中断
13:12	px6_int_type	W/R	Px[6] 中断类型及使能控制位： 00 ：不产生中断；01 ：上升沿中断； 10 ：下降沿中断；11 ：双沿中断
11:10	px5_int_type	W/R	Px[5] 中断类型及使能控制位： 00 ：不产生中断；01 ：上升沿中断； 10 ：下降沿中断；11 ：双沿中断
9:8	px4_int_type	W/R	Px[4] 中断类型及使能控制位： 00 ：不产生中断；01 ：上升沿中断； 10 ：下降沿中断；11 ：双沿中断
7:6	px3_int_type	W/R	Px[3] 中断类型及使能控制位： 00 ：不产生中断；01 ：上升沿中断； 10 ：下降沿中断；11 ：双沿中断
5:4	px2_int_type	W/R	Px[2] 中断类型及使能控制位： 00 ：不产生中断；01 ：上升沿中断； 10 ：下降沿中断；11 ：双沿中断
3:2	px1_int_type	W/R	Px[1] 中断类型及使能控制位： 00 ：不产生中断；01 ：上升沿中断； 10 ：下降沿中断；11 ：双沿中断
1:0	px0_int_type	W/R	Px[0] 中断类型及使能控制位： 00 ：不产生中断；01 ：上升沿中断； 10 ：下降沿中断；11 ：双沿中断

4.2.12 中断状态寄存器（Px_INTP_STA）(x=A..D)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
预留															
ro															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
px_int_state															
rw															

Bit	Name	W/R	Description
31:16	预留	--	预留

15:0	px_int_state	WC	中断状态寄存器,对应的 bit 位为 1 代表产生中断，否则代表无中断。中断标志置位后保持，直到向对应 bit 写 1，中断标志清除

5 CRC计算单元

5.1 CRC简介

循环冗余校验计算单元（CRC）根据选定的生产多项式得到任一16/32位的CRC计算结果。在其他的应用中，CRC技术主要应用于核实数据传输或者数据存储的正确性和完整性。

5.2 CRC主要特性

- 支持 CRC-16/32 两种 CRC 计算方式
- 支持 CRC-16 多项式：0x8005 和 0x1021
- 支持 CRC-32 多项式：0x04C11DB7
- 数据按字节输入
- 输入数据可配置由硬件进行大小端翻转
- 输出数据可配置由硬件进行大小端翻转
- 支持指定 CRC 计算初始值

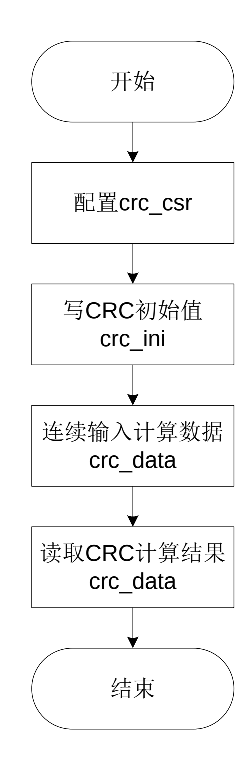
5.3 CRC功能描述

CRC单元包含一个32位的数据输入寄存器

- 对其进行写操作时，作为 8 位的输入寄存器（仅低 8 位有效）
- 对其进行读操做时，作为输出寄存器，其有效位宽由 CRC 控制状态寄存器（CRC_CSR）决定。

根据需要配置完成控制状态寄存器（CRC_CSR）和 CRC 计算初始值寄存器（CRC_INI）后对数据寄存器进行连续写操作，需要校验的数据写操作完成后，对 CRC 数据寄存器进行读操作获取 CRC 校验值。

下图为CRC操作流程图：



5.4 CRC寄存器

5.4.1 地址映射表

CRC外设基地址表

地址范围	基地址	外设	总线
0x4001_2000-0x4001_2FFF	0x4001_2000	CRC	APB0

表 5-1 CRC寄存器表

偏移地址	寄存器名称	宽度（bit）	复位值
0x00	CRC_CSR	32	0x00000000
0x04	CRC_INI	32	0x00000000
0x08	CRC_DATA_L	32	0x00000000
0x0C	CRC_DATA_H	32	0x00000000

5.4.2 控制状态寄存器（CRC_CSR）

31	30	29	28	27	26	25	24
预留							
ro							

23	22	21	20	19	18	17	16
预留							
ro							

15	14	13	12	11	10	9	8
预留							
ro							

7	6	5	4	3	2	1	0
预留			xor_out_sel	rev_out_sel	rev_in_sel	type_sel	poly_sel
ro			rw	rw	rw	rw	rw

Bit	Name	W/R	Description
31:5	预留	--	预留
4	xor_out_sel	W/R	CRC 计算结果和 0xffff 进行异或； （此步骤在 rev_out_sel 之后进行） 1：和 0xffff 进行异或； 0：计算结果直接输出。
3	rev_out_sel	W/R	CRC 计算结果高低位反转； 1：反转； 0：不反转。
2	rev_in_sel	W/R	CRC 8-bit 输入大小端反转进行计算，如 bit7 作为 bit0 参与运算，bit6 作为 bit1，以此类推。 1：反转； 0：不反转。
1	type_sel	W/R	CRC 类型选择； 1：CRC32； 0：CRC16。
0	poly_sel	W/R	CRC16 的多项式选择，选择 CRC32 时，该位无效。 1：0x1021； 0：0x8005。

5.4.3 初始值寄存器（CRC_INI）

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
crc_ini															
rw															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
crc_ini															
rw															

Bit	Name	W/R	Description
31:0	crc_ini	W/R	CRC 计算的初始值； 计算 CRC16 时，低 16 位有效。

5.4.4 数据寄存器（CRC_DATA）

CRC_DATA_IN输入（写操作）

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
预留															
ro															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
预留								crc_data							
ro								wo							

Bit	Name	W/R	Description
31:8	预留	--	预留
7:0	crc_data	WO	CRC 数据输入 作为数据输入寄存器仅低 8 位有效

CRC_DATA_OUT输出（读操作）

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
crc_data															
ro															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
crc_data															
ro															

Bit	Name	W/R	Description
31:0	crc_data	RO	CRC 计算结果； 计算 CRC16 时，低 16 位有效。

6 真随机数发生器（TRNG）

6.1 TRNG简介

TRNG单元用于产生真随机数序列。

6.2 TRNG主要特性

- 一次工作产生 128-bit 真随机数序列；
- 可配置随机数生成后产生 CPU 中断请求；

6.3 TRNG功能描述

① 配置控制状态寄存器（TRNG_CSR）及模拟控制寄存器（TRNG_ANA）
② 将TRNG_CSR[s128]清“0”，TRNG单元开始产生随机数
③ 轮询TRNG_CSR[s128]，硬件置“1”表示随机数生成完成（中断模式中，随机数生成后产生中断）。
④ 连续读TRNG_DATA 4次获取128bit随机数
⑤ 循环②~④获取更多随机数

6.4 TRNG寄存器

6.4.1 地址映射表

TRNG外设基地址表

地址范围	基地址	外设	总线
0x4001_E000-0x4001_EFFF	0x4001_E000	TRNG	APB0

表 6-1 TRNG寄存器表

偏移地址	寄存器名称	宽度（bit）	复位值
0x00	TRNG_CSR	32	0x000000A0
0x04	TRNG_DATA	32	0x00000000
0x0C	TRNG_ANA	32	0x40000807
0x10	TRNG_PN	32	0x69D84C18
0 x14	RNG_INDEX	32	0x00000000

6.4.2 控制状态寄存器（RNG_CSR）

31	30	29	28	27	26	25	24
预留							
ro							

23	22	21	20	19	18	17	16
预留							
ro							

15	14	13	12	11	10	9	8
预留							
ro							

7	6	5	4	3	2	1	0
预留		edge_sel	intp_en	预留	rng0_attack	预留	rng0_s128
ro		rw	rw	ro	rw	ro	rw

Bit	Name	W/R	Description
31:6	预留	--	预留
5	edge_sel	W/R	RNG数据采样沿选择。 1：clk_ana下降沿采样； 0：clk_ana上升沿采样（SMCU应选择上升沿）。
4	intp_en	W/R	0：不产生中断； 1：产生中断。
3	预留	--	预留
2	rng0_attack	W/R	RNG0 1:检测到连续47个0或1，有攻击 0:没有攻击
1	预留	--	预留
0	rng0_s128	W/R	RNG0,128-bit随机数状态位，硬件置1软件清0，清0后自动开始新的128-bit随机数生成： 0：128-bit随机数未生成； 1：128-bit随机数已生成。

6.4.3 数据寄存器（RNG_DATA）

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
data															
ro															
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
data															
ro															

Bit	Name	W/R	Description
31:0	data	R	32-bit 随机数，需在 s128 置 1 后读取，连续 4 次读取该寄存器，来获取 128-bit 随机数。

6.4.4 模拟控制寄存器（RNG_ANA）

31	30	29	28	27	26	25	24
vnc_bypass	pd_trng0	lfsr_xor_fb	ana_out_en	预留			
rw	rw	rw	rw	ro			
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
预留							
ro							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
预留							
ro							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
预留							
ro							

Bit	Name	W/R	Description
31	vnc_bypass	RW	1:冯诺依曼Bypass 0:冯诺依曼开启
30	pd_trng0	RW	模拟随机数0的PD信号；

29	lfsr_xor_fb	RW	1:将LFSR的输出和模拟的输出进行异或反馈给LFSR； 0:LFSR的反馈仅为自身反馈
28	ana_out_en	RW	RNG0 1:直接模拟输出 0:经数据后处理输出
27:0	预留	--	预留

6.4.5 伪随机序列寄存器（RNG_PN）

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
pn															
rw															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
pn															
rw															

Bit	Name	W/R	Description
31:0	pn	RW	32-bit 伪随机序列，1 个系统 cycle 更新一次。 该寄存器初始值可配。

6.4.6 RNG FIFO Index

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
fifo_rd_ov	预留														
wc	ro														

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
预留														rng0_index	
ro														ro	

Bit	Name	W/R	Description
31	fifo_rd_ov	WC	fifo读溢出标示，当读取完新生成的128bit随机数后，再对fifo进行读操作，该位置1，软件写操作清0
30:2	预留	--	预留
1:0	rng0_index	RO	RNG0 FIFO已读深度

7 OTP控制模块（OTP_CTRL）

7.1 OTP简介

OTP是1块具有单次写操作的特殊存储器，OTP出厂时内部数据经过初始化后bit位均为“0”，OTP写操作只能将内部bit位由“0”写为“1”，而不能由“1”改为“0”。

OTP开放区域为0x4000_8180~0x4000_81FF

7.2 OTP功能描述

7.2.1 OTP只读锁定

OTP提供区域写保护和区域写保护锁定功能。

区域写保护：

OTP区域写保护bit位为“0”时，对应区域可以进行编程/擦除操作，为“1”时，对应区域只能进行读操作

区域写保护锁定：

OTP区域写保护锁定bit位为“0”时，对应区域写保护bit位可以修改，为“1”时，对应区域写保护bit位保持已有状态不可修改。

7.2.2 OTP编程操作保护

为防止用户程序对OTP的误操作，OTP在启动编程/擦除操作时，需要进行固定的寄存器操作后，再启动编程/擦除操作使能。

编程/擦除操作前需要对OTP_PROT进行2次连续写操作，第1次写入：0xABCD00A5，第2次写入：0x1234005A。2次写入操作完成后，紧接着进行启动编程/擦除操作使能，如果期间有其他FCU操作则启动编程/擦除操作使能视为无效。

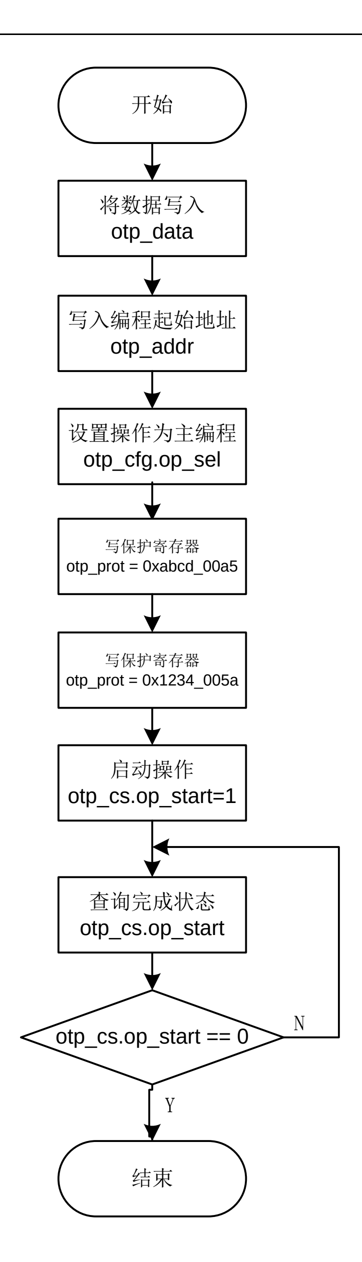
操作流程如下：

OTP_PROT = 0xABCD00A5;

OTP_PROT = 0x1234005A;

完成以上2次寄存器操作后，启动编程/擦除操作使能

7.2.3 OTP编程操作



7.3 OTP_CTRL寄存器

7.3.1 地址映射表

寄存器基地址表

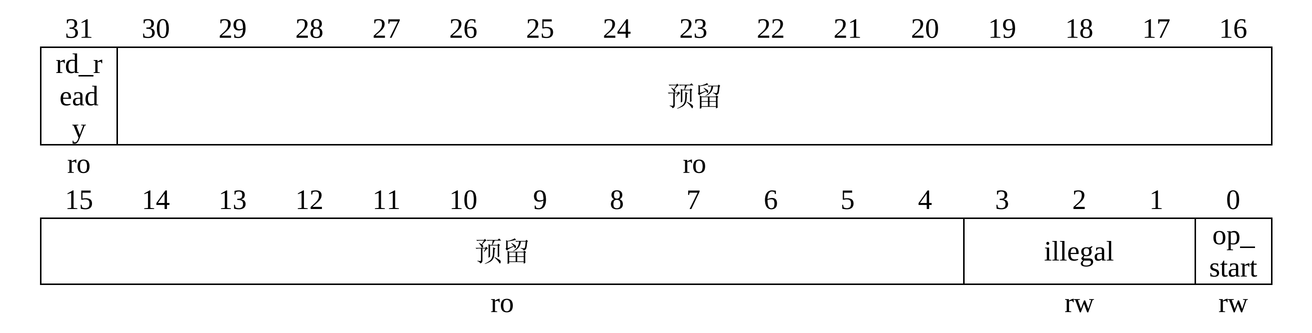
地址范围	基地址	外设	总线
0x4000_8000-0x4000_BFFF	0x4000_8000	OTP_CTRL	AHB

表 7-1 OTP_CTRL寄存器表

偏移地址	寄存器名称	宽度（bit）	复位值
0x0000-0x01FF	OTP 数据区	32	0x00000000
0x0200-0x2000	保留	32	0x00000000
0x2004	OTP_CS	32	0x80000000
0x2008	OTP_PROT	32	0x00000000
0x200C	OTP_ADDR	32	0x00000000
0x2010	OTP_PDATA	32	0x00000000
0x2014	OTP_RO	32	0xFFFFFFFF
0x2018	OTP_ROL	32	0x00000000
0x201C	保留	32	0x00000000

0x2020	OTP_TIM	32	0x00000132
0x2024	OTP_TIM_EN	32	0x00000000
0x2028	OTP_RO1	32	0xFFFFFFFF
0x202C	OTP_ROL1	32	0x00000000
0x2030	OTP_RO2	32	0xFFFFFFFF
0x2034	OTP_ROL2	32	0x00000000
0x2038	OTP_RO3	32	0xFFFFFFFF
0x203C	OTP_ROL3	32	0x00000000
0x2040	OTP_TMRF	32	0x8000FFFF

7.3.2 OTP控制状态寄存器（OTP_CS）



Bit	Name	W/R	Description
31	rd_ready	RO	1：可进行OTP读操作 0：OTP处于编程/休眠状态，不可进行读操作
30:4	预留	-	保留位
3:1	illegal	RW	操作完成状态，硬件置位软件清0： 000：此次操作无异常； 001：在编程/休眠状态下对OTP进行读操作 010：对只读区域进行编程 011：编程范围超出OTP地址范围 100：在休眠状态下进行编程操作 101：在非休眠状态下进行唤醒
0	op_start	RW	启动对OTP的操作，软件置1硬件清0，软件可通过该位查询操作是否完成。

7.3.3 OTP启动保护寄存器（OTP_PROT）

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
prot															
wc															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
prot															
wc															

Bit	Name	W/R	Description
31:0	prot	WC	启动保护控制位。 要启动 OTP 编程，需进行 3 次连续的写操作，顺序如下：写 prot 0xabcd_00a5，写 prot 0x1234_005a，将 op_start 置 1。

7.3.4 OTP编程擦除地址寄存器（OTP_ADDR）

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
op_addr															
rw															
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
op_addr															
rw															

Bit	Name	W/R	Description
31:0	op_addr	RW	编程地址

7.3.5 OTP编程数据寄存器（OTP_PDATA）

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
预留															
ro															
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
预留								pdata							
ro								rw							

Bit	Name	W/R	Description
31:8	预留	--	预留
7:0	pdata	W/R	用于存放 8bit 数据编程数据。 只能在 op_start 为 0 时可写。

7.3.6 OTP主存区域只读区域寄存器（OTP_RO）

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ro															
rw															
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
ro															
rw															

Bit	Name	W/R	Description
31:0	ro	RW	按 32bit 将 OTP 前 128 字节划分成 32 个区域，每个区域由 ro 的 1bit 控制。 0：可编程。 1：只读。

7.3.7 OTP主存区只读锁定寄存器（OTP_ROL）

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ro_lock															
rw															
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
ro_lock															
rw															

Bit	Name	W/R	Description
31:0	ro_lock	W/R	用于锁定 OTP_RO 寄存器的配置，一旦锁定，otp_ro 对应位不能修改，复位后锁定解除。 0：不锁定。 1：锁定，置 1 后软件无法清 0，只有复位后硬件清 0。 ro_lock[0]对应 ro[0]，ro_lock[1]对应 ro[1]，以此类推。

7.3.8 OTP时序寄存器（OTP_TIM）

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
预留															
ro															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
预留					cycles_10ns			cycles_1us							
ro					rw			rw							

Bit	Name	W/R	Description
31:11	预留	--	保留位
10:8	cycles_10ns	RW	决定10ns所用的时钟周期，比如60M时，周期为16.67ns，该寄存器写1。 只能在op_start为0时可写。
7:0	cycles_1us	RW	决定1us所用的时钟周期，比如60M时，周期为16.67ns，为保险起见减去1ns作为余量，所以该寄存器配置为1000ns/15ns 有余数则加1，得 67。 只能在op_start为0时可写。

7.3.9 OTP时序使能寄存器（OTP_TIM_EN）

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
预留															
ro															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
预留								tim_en							
ro								rw							

Bit	Name	W/R	Description
31:8	预留	-	保留位
7:0	tim_en	W/R	当该寄存器等于A5时，使用otp_tim作为时序参考，否则由系统自动生成

7.3.10 OTP主存区域只读区域寄存器1（OTP_RO1）

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ro															
rw															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ro															

rw

Bit	Name	W/R	Description
31:0	ro	RW	按 32bit 将 OTP 128~256 字节划分成 32 个区域，每个区域由 ro 的 1bit 控制。 0：可编程。 1：只读。

7.3.11 OTP主存区只读锁定寄存器1（OTP_ROL1）

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ro_lock															
rw															
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
ro_lock															
rw															

Bit	Name	W/R	Description
31:0	ro_lock	W/R	用于锁定 OTP_RO1 寄存器的配置，一旦锁定，otp_ro1 对应位不能修改，复位后锁定解除。 0：不锁定。 1：锁定，置 1 后软件无法清 0，只有复位后硬件清 0。 ro_lock[0]对应 ro[0]，ro_lock[1]对应 ro[1]，以此类推。

7.3.12 OTP主存区域只读区域寄存器2（OTP_RO2）

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ro															
rw															
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
ro															
rw															

Bit	Name	W/R	Description
31:0	ro	RW	按 32bit 将 OTP 256~384 字节划分成 32 个区域，每个区域由 ro 的 1bit 控制。 0：可编程。 1：只读。

7.3.13 OTP主存区只读锁定寄存器2（OTP_ROL2）

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ro_lock															
rw															
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
ro_lock															
rw															

Bit	Name	W/R	Description
31:0	ro_lock	W/R	用于锁定 OTP_RO2 寄存器的配置，一旦锁定，otp_ro2 对应位不能修改，复位后锁定解除。 0：不锁定。 1：锁定，置 1 后软件无法清 0，只有复位后硬件清 0。 ro_lock[0]对应 ro[0]，ro_lock[1]对应 ro[1]，以此类推。

7.3.14 OTP主存区域只读区域寄存器3（OTP_RO3）

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ro															
rw															
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
ro															
rw															

Bit	Name	W/R	Description
31:0	ro	RW	按 32bit 将 OTP 384~512 字节划分成 32 个区域，每个区域由 ro 的 1bit 控制。 0：可编程。 1：只读。

7.3.15 OTP主存区只读锁定寄存器3（OTP_ROL3）

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ro_lock															
rw															
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
ro_lock															
rw															

Bit	Name	W/R	Description
31:0	ro_lock	W/R	用于锁定 OTP_RO3 寄存器的配置，一旦锁定，otp_ro3 对应位不能修改，复位后锁定解除。 0：不锁定。 1：锁定，置 1 后软件无法清 0，只有复位后硬件清 0。 ro_lock[0]对应 ro[0]，ro_lock[1]对应 ro[1]，以此类推。

7.3.16 OTP时间基准（OTP_TMRF）

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
fix_sel	预留														
rw	ro														
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
fix_time															

rw

Bit	Name	W/R	Description
31	fix_sel	RW	1：使用fix_time作为OTP编程电压启动等待时间 0：OTP编程电压启动等待时间100us
30:16	预留	--	预留
15:0	fix_time	RW	以时钟周期为单位进行技术，例如fix_time为8，OTP编程电压启动等待8个时钟周期

8 CACHE模块（CACHE）

8.1 CACHE简介

Cache用于提升处理器从低速存储器中取指的效率，为两者的中间媒介。其原理是根据程序局部性原则，通过小容量速度快的存储器缓存部分指令或数据，以减少处理器对慢速大容量存储器的访问次数，从而提升处理器效率。

8.2 CACHE寄存器描述

8.2.1 地址映射表

寄存器基地址表

地址范围	基地址	外设	总线
0x4008_0000-0x4008_FFFF	0x4008_0000	CACHE_CTRL	AHB

表 8-1 CACHE寄存器表

偏移地址	寄存器名称	宽度（bit）	复位值
0x00	CACHE_I0	32	0x00000000
0x04	CACHE_I1	32	0x00000000
0x08	CACHE_I2	32	0x00000000
0x0C	CACHE_I3	32	0x00000000
0x10	CACHE_K0	32	0x00000000
0x14	CACHE_K1	32	0x00000000
0x18	CACHE_K2	32	0x00000000
0x1C	CACHE_K3	32	0x00000000
0x20	CACHE_CS	32	0x00000000
0x24	CACHE_REF	32	0x00000000
0x28-0x3C	保留	32	0x00000000
0x40	CACHE_CONFIG	32	0x00000000
0x44-0x70	保留	32	0x00000000
0x74	CACHE _SADDR	32	0x00000000
0x78	CACHE _EADDR	32	0x00000000

8.2.2 向量寄存器（CACHE_Ix）（x=0…3）

- **Name:** CACHE_Ix
- **Size:** 32 bits
- **Address Offset:**
  for x = 0, 0x00
  for x = 1, 0x04
  for x = 2, 0x08
  for x = 3, 0x0C
- **Read/write access:** write only

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ivx															
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
ivx															

Bit	Name	W/R	Description

31:0	ivx	WO	初始向量寄存器 x（x=0…3），key_gen 必须为 A5 时该寄存器可写

8.2.3 密钥寄存器（CACHE_Kx）（x=0…3）

- **Name:** CACHE_Kx
- **Size:** 32 bits
- **Address Offset:**
  for x = 0, 0x10
  for x = 1, 0x14
  for x = 2, 0x18
  for x = 3, 0x1C
- **Read/write access:** write only

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
keyx															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
keyx															

Bit	Name	W/R	Description
31:0	keyx	WO	密钥寄存器 x（x=0…3），key_gen 必须为 A5 时该寄存器可写

8.2.4 控制寄存器（CACHE_CS）

- **Name:** CACHE_CS
- **Size:** 32 bits
- **Address Offset:**0x20
- **Read/write access:** read/write

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
key_gen_start	key_gen_err	cache_busy	预留												
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
预留								key_gen							

Bit	Name	W/R	Description
31	key_gen_start	RW	置1后启动轮密钥生成，软件置1，完成轮密钥生成后硬件清0（key_gen必须为A5，且cache不在解密状态，该位才能被硬件允许置1）。
30	key_gen_err	RW	0:无密钥生成错误 1:密钥生成不能进行，如下两种情况导致， 1）key_gen不等于A5， 2）cache正在对数据进行解密时，启动密钥生成 硬件置1，软件清0
29	cache_busy	RO	1：cache正在从Flash中取指 0：cache没有从Flash中取指
28:8	预留	--	保留位
7:0	key_gen	RW	等于A5：密钥生成模式，KEY/IV可写 不等于A5：解密模式，KEY/IV不可写

8.2.5 CACHE刷新控制寄存器（CACHE_REF）

- **Name:** CACHE_REF
- **Size:** 32 bits
- **Address Offset:**0x24
- **Read/write access:** read/write

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
refresh	all_tag	预留						refresh_index							

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
refresh_index															

Bit	Name	W/R	Description
31	refresh	RW	置1后刷新refresh_index指定Cache TAG，刷新完成后自动清0
30	all_tag	RW	0:只根据refresh_index进行刷新 1:对所有TAG进行进行全刷新 软件置1硬件清0
29:24	预留	--	保留位
23:0	refresh_index	RW	刷新地址，对应CODE BUS地址的23:0bit

8.2.6 CACHE配置寄存器（CACHE_CONFIG）

- **Name:** CACHE_CONFIG
- **Size:** 32 bits
- **Address Offset:**0x40
- **Read/write access:** read/write

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
sect_enc								wrap_ctrl							

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
预留								bypass							

Bit	Name	W/R	Description
31:24	sect_enc	RW	当等于A5时，且bypass关闭，则开启部分区域的解密功能 不等于A5时，且bypass关闭，则开启全区域的解密功能
23:16	wrap_ctrl	RW	当等于A5时， 对QSPI Controller的AHB WRAP读取开启 不等于A5时， 对QSPI Controller的AHB WRAP读取关闭
15:8	预留	--	保留位
7:0	bypass	RW	等于A5：解密bypass 不等于A5：解密模式

8.2.7 区域加密起始地址寄存器（CACHE _SADDR）

- **Name:** CACHE _SADDR
- **Size:** 32 bits
- **Address Offset:** 0x74
- **Read/write access:** read/write

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
saddr															
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
saddr															

Bit	Name	W/R	Description
31:0	saddr	RW	区域加密起始地址，当 saddr<读取地址<eaddr，对 Flash 读取数据进行解密操作

8.2.8 区域加密结束地址寄存器（CACHE _EADDR）

- **Name:** CACHE _EADDR
- **Size:** 32 bits
- **Address Offset:** 0x78
- **Read/write access:** read/write

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
eaddr															
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
eaddr															

Bit	Name	W/R	Description
31:0	eaddr	RW	区域加密结束地址，当 saddr<读取地址<eaddr，对 Flash 读取数据进行解密操作

9 备份寄存器（BPK）

9.1 BPK简介

BPK、RTC、SENSOR都处于电池电源域。电池和主电源同时存在的情况下，由主电源和电池电源同时给安全区供电，此时两个电源中电压偏高的电源提供较大的电流。

系统提供32 x 16的带电存储区域（BPK），其数据在受到攻击后由硬件自动清除。

9.2 BPK特性

- BPK内数据在受到攻击后由硬件清除
- 支持按区域锁定或清除，区域大小32*16

9.3 BPK寄存器

9.3.1 地址映射表

BPK基地址表

地址范围	基地址	外设	总线
0x4003_0000-0x4003_00A0	0x 4003_0000	BPK	APB2

表 9-1 BPK寄存器表

偏移地址	寄存器名称	宽度（bit）	复位值
0x0000	BPK0	32	0x00000000
0x0004	BPK1	32	0x00000000
0x0008	BPK2	32	0x00000000
… …	… …	32	0x00000000
0x003C	BPKn	32	0x00000000
0x0040-0x007C	RSVD0	32	0x00000000
0x0080	BPK_RDY	32	0x00000003
0x0084	BPK_CLR	32	0x00000000
0x0088	BPK_LRA	32	0x00000000
0x008C	BPK_LWA	32	0x00000000
0x0090	保留	32	0x00000000
0x0094	BPK_LR	32	0x00000000
0x0098	BPK_SCR	32	0x00000000
0x009C	BPK_POWER	32	0x00000001

9.3.2 BPK寄存器（BPKx）（x=0..15）

偏移地址：0x0000 – 0x003C　　复位值：0x00000000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
bpkx															
rw															
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
bpkx															

rw

Bit	Name	W/R	Description
31:0	bpkx	W/R	备份寄存器 x，x = 0..15（密钥寄存器）

9.3.3 BPK状态寄存器（BPK_RDY）

偏移地址：0x0080 复位值：0x00000003

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
预留															
ro															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
预留														bpk_por	bpk_rdy
ro														rw	ro

Bit	Name	W/R	Description
31:2	预留	RO	预留
1	bpk_por	W/R	1：表明 BPK 刚经历掉电复位或 bpk_rr 复位； 硬件置 1 软件清 0
0	bpk_rdy	RO	每次“电池电源域上电”或“探测到攻击复位 RTC”之后，软件对密钥区进行操作前，需查询 bpk_rdy 状态， 0：表明 BPK 正在复位，无法对密钥区进行操作； 1：表明 BPK 复位结束，可对密钥区进行读写操作.

9.3.4 密钥清除寄存器（BPK_CLR）

偏移地址：0x0084 复位值：0x00000000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
预留															
ro															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
预留														bpk_clr1	bpk_clr0
ro														wc	wc

Bit	Name	W/R	Description
31:2	预留	--	预留
1	bpk_clr1	WC	清除密钥区域 1
0	bpk_clr0	WC	清除密钥区域 0

9.3.5 密钥锁定寄存器（BPK_LRA）

偏移地址：0x0088 复位值：0x00000000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
预留															
ro															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

预留	bpk_lra1	bpk_lra0
ro	rw	rw

Bit	Name	W/R	Description
31:2	预留	--	预留
1	bpk_lra1	RW	密钥区 1 读锁定 1:锁定，不可读 0:可读
0	bpk_lra0	RW	密钥区 0 读锁定 1:锁定，不可读 0:可读

9.3.6 密钥锁定寄存器（BPK_LWA）

偏移地址：0x008C 复位值：0x00000000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
预留															
ro															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
预留														bpk_lwa1	bpk_lwa0
ro														rw	rw

Bit	Name	W/R	Description
31:2	预留	--	预留
1	bpk_lwa1	RW	密钥区 1 写锁定 1:锁定，不可写 0:可写
0	bpk_lwa0	RW	密钥区 0 写锁定 1:锁定，不可写 0:可写

9.3.7 密钥锁定寄存器（BPK_LR）

偏移地址：0x0094 复位值：0x00000000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
预留															
ro															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
预留										bpk_lscr	bpk_lclr	bpk_llra	bpk_llwa	bpk_rr	bpk_lr
ro										rw	rw	rw	rw	rw	rw

Bit	Name	W/R	Description
31:6	预留	--	预留
5	bpk_lscr	RW	1:锁定 bpk_scr 0:不锁定 bpk_scr

4	bpk_lclr	RW	1:锁定 bpk_clr 0:不锁定 bpk_clr
3	bpk_llra	RW	1:锁定 bpk_lra 0:不锁定 bpk_lra
2	bpk_llwa	RW	1:锁定 bpk_lwa 0:不锁定 bpk_lwa
1	bpk_rr	RW	1:锁定 bpk_rr 0:不锁定 bpk_rr
0	bpk_lr	RW	1:锁定 bpk_lr 0:不锁定 bpk_lr 备注：锁定 bpk_lr 后只能通过 BPK_RR 或 VBAT 掉电复位解锁

9.3.8 密钥加扰寄存器（BPK_SCR）

偏移地址：0x0098　　　　复位值：0x00000000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
bpk_scr															
wo															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
bpk_scr															
wo															

Bit	Name	W/R	Description
31:0	bpk_scr	WO	密钥加扰寄存器，存放扰码

10 实时时钟（RTC）

10.1 RTC简介

实时时钟是一个独立的定时器。RTC模块拥有一组连续计数的计数器。 RTC模块和RTC相关配置寄存器都处于电池电源域，即主电源掉电对RTC没有任何影响，RTC依旧保持正常计数。

10.2 RTC特性

- 以秒作为计时单位（通过配置产生秒中断）
- CPU独立中断源
- 32位闹钟设置寄存器，用于在设定时间处产生中断信号或唤醒CPU
- RTC实时时钟单元，支持报警中断生成

10.3 RTC寄存器

10.3.1 地址映射表

RTC基地址表

地址范围	基地址	外设	总线
0x4003_0000 - 0x4003_00B8	0x4003_00A0	RTC	APB2

表 10-1 RTC寄存器表

偏移地址	寄存器名称	宽度（bit）	复位值
0x00A0	RTC_CS	32	0x00000008
0x00A4	RTC_REF	32	0x00000000
0x00A8	RTC_ARM	32	0x0000FFFF
0x00AC	RTC_TIM	32	0x00000000
0x00B0	RTC_INTCLR	32	0x00000000
0x00B4	OSC32k_CR	32	0x00000060
0x00B8	RTC_ATTA_TIM	32	0x00000000

10.3.2 RTC控制状态寄存器（RTC_CS）

偏移地址：0x00A0 复位值：0x00000008

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
预留															
ro															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
预留											rtc_clr	rtc_rdy	rtc_ien	time_rd_lock	intp_rd_bit
ro											rw	ro	rw	rw	ro

Bit	Name	W/R	Description
31:5	预留	--	预留位
4	rtc_clr	W/R	0：RTC正常计数，可从rtc_tim寄存器读取当前计数值；

			1：RTC计数器清0，即rtc_tim寄存器清0。 软件置1硬件清0，即上电后RTC就处于计数状态。
3	rtc_rdy	RO	每次"电池电源域上电"软件对RTC进行操作前，需查询rtc_rdy状态， 0：表明RTC正在复位，无法对RTC进行操作； 1：表明RTC复位结束，可对RTC进行读写操作，
2	rtc_ien	W/R	RTC中断使能。 0：关闭中断； 1：使能中断。
1	time_rd_lock	W/R	RTC当前值读取锁定位，当CPU要读取当前RTC的计数值时，需要先将该为置1，读取当前值后需要将该位置0。
0	intp_rd_bit	RO	RTC中断标识，只有中断使能后，该位才能置1 0：无中断； 1：有中断。

10.3.3 RTC计数初始值寄存器（RTC_REF）

偏移地址：0x00A4　　复位值：0x00000000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
rtc_ref															
rw															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
rtc_ref															
rw															

Bit	Name	W/R	Description
31:0	rtc_ref	W/R	RTC 计时初始值寄存器，仅用于保存计时初始值，不参加计时。 实际计时值=rtc_ref+rtc_tim；

10.3.4 RTC闹钟设置寄存器（RTC_ARM）

偏移地址：0x00A8　　复位值：0x0000FFFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
rtc_arm															
rw															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
rtc_arm															
rw															

Bit	Name	W/R	Description
31:0	rtc_arm	W/R	RTC闹钟设置寄存器，当rtc_tim=rtc_arm时产生中断；

10.3.5 RTC当前计数值寄存器（RTC_TIM）

偏移地址：0x00AC　　复位值：0x00000000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
rtc_tim															

ro															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
rtc_tim															
ro															

Bit	Name	W/R	Description
31:0	rtc_tim	RO	RTC 32bit计数器的当前计数值。读取该寄存器前需将rtc_cs.time_rd_lock置1，读取完后需置0。 实际计时值=rtc_ref+rtc_tim。

10.3.6 RTC中断清除寄存器（RTC_INTCLR）

偏移地址：0x00B0 复位值：0x00000000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
预留															
ro															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
预留															rtc_intclr
ro															wc

Bit	Name	W/R	Description
31:1	预留	--	预留位
0	rtc_intclr	WC	对该寄存器进行写操作可清除中断。

10.3.7 32K时钟校准控制寄存器（OSC32K_CR）

偏移地址：0x00B4 复位值：0x00000060

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
预留															
ro															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
预留								osc32k_cal_done	osc32k_cal_en	osc32k_cal_word					
ro								rc	rw	rw					

Bit	Name	W/R	Description
31:8	预留	--	预留
7	osc32k_cal_done	RC	32KHz振荡器校准完成标识
6	osc32k_cal_en	RW	32KHz振荡器校准使能，当校准完成后，此位自动清零；
5:0	osc32k_cal_word	RW	32KHz振荡器校准控制字；

10.3.8 RTC攻击时刻记录寄存器（RTC_ATTA_TIM）

偏移地址：0x00B8 复位值：0x00000000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
rtc_atta_tim															

ro

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
rtc_atta_tim															
ro															

Bit	Name	W/R	Description
31:0	rtc_atta_tim	RO	当攻击产生时，由该寄存器记录当时RTC的时间

11 传感器单元（SENSOR）

11.1 SENSOR简介

BPK、RTC、SENSOR都处于电池电源域。电池和主电源同时存在的情况下，由主电源和电池电源同时给安全区供电，此时两个电源中电压偏高的电源提供较大的电流。

11.2 SENSOR特性

- 可配外部动态/静态TAMPER
- 最多8路外部静态TAMPER
- 高、低温检测
- 高、低压检测
- Active shielding
- 32K时钟检测
- 12M时钟检测
- 探测到攻击后产生中断/复位选择
- 动态Tamper/active shielding翻转频率设定

11.3 外部静态/动态功能说明

外部传感器可配成静态或动态模式，静态传感器的“攻击电平”固定，外部传感器检测到攻击电平后擦除BPK中的数据；

“静态传感器”指正常情况下，传感器的输入端口为固定的0或1，当该电平翻转后，则传感器认为发生攻击。

静态配置时，各端口均为输入，外部传感器对端口的高低电平进行检测。当检测到“攻击电平”时，激活“BPK擦除操作”。各端口对应传感器如下表。

表 11-1 静态传感器端口表

传感器名	端口（方向均为输入）
静态传感器 0	ext_s0(高电平产生攻击，内部上拉默认开启)
静态传感器 1	ext_s1(高电平产生攻击，内部上拉默认开启)
静态传感器 2	ext_s2(低电平产生攻击，内部下拉默认开启)
静态传感器 3	ext_s3(低电平产生攻击，内部下拉默认开启)
静态传感器 4	ext_s4(高电平产生攻击，内部上拉默认开启)
静态传感器 5	ext_s5(高电平产生攻击，内部上拉默认开启)
静态传感器 6	ext_s6(低电平产生攻击，内部下拉默认开启)
静态传感器 7	ext_s7(低电平产生攻击，内部下拉默认开启)

“动态传感器”指传感器的输出输入端口组成回环，输出端口发送随机数（输出频率可配），输入端口接收，如果发送和接收数据不相同，则认为发生攻击。

“动态传感器”相邻两端口组成为动态传感器的输出/输入端口，各端口对应传感器如表“动态传感器端口”所示。

表 11-2 动态传感器端口表

传感器名	输出	输入
动态传感器 0	ext_s0	ext_s1
动态传感器 1	ext_s2	ext_s3
动态传感器 2	ext_s4	ext_s5
动态传感器 3	ext_s6	ext_s7

11.4 SENSOR寄存器

11.4.1 地址映射表

SENSOR基地址表

地址范围	基地址	外设	总线
0x4003_0000-0x4003_0158	0x4003_000C0	SENSOR	APB2

表 11-3 SENSOR寄存器表

偏移地址	寄存器名称	宽度（bit）	复位值
0x00BC	BPK_RR	32	0x00000000
0x00C0	SEN_EXT_TYPE	32	0x000FF000
0x00C4	SEN_EXT_CFG	32	0x00A5A000
0x00C8	SEN_EXT_EN	32	0x00000000
0x00CC	SEN_STATE	32	0x00000000
0x00D0	SEN_RSVD0	32	0x00000000
0x00D4	SEN_ANA_EN	32	0x000000F0
0x00D8	SEN_ANA_THR	32	0x80000000
0x00DC	SEN_ATTACK_CNT	32	0x0000000F
0x00E0	SEN_ATTACK_TYP	32	0x00000001
0x00E4	SEN_VG_DETECT	32	0x00000000
0x00E8	SEN_RNG_INI	32	0x00000000
0x00EC-0x0100	SEN_RSVD1	32	0x00000000
0x0104	SEN_EXT0_EN	32	0x000000AA
0x0108	SEN_EXT1_EN	32	0x000000AA
0x010C	SEN_EXT2_EN	32	0x000000AA
… …	… …	32	0x000000AA
0x0120	SEN_EXT7_EN	32	0x000000AA
0x0124-0x0130	SEN_RSVD2	32	0x000000AA
0x0134	SEN_VH_EN	32	0x000000AA
0x0138	SEN_VL_EN	32	0x000000AA
0x013C	SEN_TH_EN	32	0x000000AA
0x0140	SEN_TL_EN	32	0x000000AA
0x0144	SEN_XTAL32_EN	32	0x000000AA
0x0148	SEN_MESH_EN	32	0x80000055
0x014C	SEN_VOLGLTCH_EN	32	0x000000AA
0x0150	SEN_EXTS_START	32	0x000000AA
0x0154	SEN_EXTS_LOCK	32	0x00000000
0x0158	SEN_ANA0	32	0x03500220
0x015C	SEN_ANA1	32	0x00000024
0x0160	SEN_ATTCLR	32	0x00000000
0x0164-0x0170	SEN_RSVD3	32	0x00000000
0x0174	SEN_PULL_MASK	32	0xFF0000FF

11.4.2 软复位寄存器（BPK_RR）

偏移地址：0x00BC 复位值：0x00000060

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
预留															
ro															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
预留															bpk_rr
ro															

Bit	Name	W/R	Description
31:1	预留	RO	
0	bpk_rr	WC	复位BPK_LR、BPK_LWA、BPK_LRA及SENSOR的所有寄存器，并清除密钥。软件置1，硬件清零

11.4.3 SEN类型配置寄存器（SEN_EXT_TYPE）

偏移地址：0x00C0 复位值：0x000FF000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
预留												pullup_en			
ro												rw			

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
pullup_en				预留				exts_type							
rw				ro				rw							

Bit	Name	W/R	Description
31:20	预留	RO	预留位
19:12	pullup_en	RW	配置8个端口的拉电阻使能，动态传感器不需使能拉电阻 0：不使能拉电阻； 1：使能拉电阻。
11:8	预留	RO	预留位
7:0	exts_type	RW	选择两个外部传感器端口对应的传感器类型；exts_type[1:0]对应ext_s0和ext_s1，exts_type[3:2]对应ext_s2和ext_s3，依次类推，可参照表 11-1和表 11-2。 0：2个端口都是静态传感器； 1：动态传感器：1个端口为动态传感器输出，另一个端口为输入

11.4.4 SEN各参数配置寄存器（SEN_EXT_CFG）

偏移地址：0x00C4 复位值：0x00A5A000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
预留					exts_att_pu	预留	exts_pupu_t	exts_gf_dyn		exts_pupu		exts_pupu_en	exts_gf_en	exts_gf	
ro					rw	ro	rw	rw		rw		rw	rw	rw	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
exts_proc	预留	freq		exts_level											
rw	ro	rw		ro											

Bit	Name	W/R	Description
31:27	预留	RO	预留位，读取值为0。
26	exts_att_pu	RW	1：使能Sensor在检测到攻击时上拉 0：检测到攻击时保持原状态上拉
25	预留	RO	预留位，读取值为0。

24	exts_pupu_t	RW	选择静态传感器上拉间隔时间T，和静态Sensor毛刺滤除的采样周期； 0：500ms输出一次； 1：1s输出一次；（实际为0.999ms）
23:22	exts_gf_dyn	RW	选择动态Tamper毛刺滤除采样次数： 00:采样1次，有攻击，则上报攻击 01:采样2次，2次均为攻击，则上报攻击 10:采样3次，3次均为攻击，则上报攻击 11:采样4次，4次均为攻击，则上报攻击 动态Sensor在信号两次翻转间隔等分成四段，对应四次采样
21:20	exts_pupu	RW	选择脉冲上拉模式的上拉持续时间 00：1/2 * (exts_pupu_t) 01：1/4 * (exts_pupu_t) 10：1/8 * (exts_pupu_t) 11：1/16 * (exts_pupu_t)
19	exts_pupu_en	RW	脉冲上拉模式总使能，使能后，将对pullup_en使能的上拉进行间隔开启关闭，关闭后，pullup_en使能的上拉将常开； 1：脉冲上拉模式开启 0：脉冲模式关闭
18	exts_gf_en	RW	1:打开毛刺滤除功能 0:关闭毛刺滤除功能
17:16	exts_gf	RW	选择静态Tamper毛刺滤除采样次数： 00:采样1次，有攻击，则上报攻击 01:采样2次，2次均为攻击，则上报攻击 10:采样3次，3次均为攻击，则上报攻击 11:采样4次，4次均为攻击，则上报攻击 静态Sensor则根据exts_pupu_t设置的间隔采样
15	exts_proc	RW	选择各传感器探测攻击后是产生中断还是复位； 0：产生复位； 1：产生中断。
14	rsvd	RO	预留位
13:12	freq	RW	选择动态传感器/Mesh的输出频率； 00：31.25ms输出一次； 01：125ms输出一次； 10：500ms输出一次； 11：1s输出一次；（实际为0.999ms）
11:0	rsvd	RO	保留位

11.4.5 SEN使能寄存器（SEN_EXT_EN）

偏移地址：0x00C8 复位值：0x00000000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
预留															
ro															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
预留															soft_attack_en
ro															rw

Bit	Name	W/R	Description
31:1	预留	RO	预留位，读取值为0。
0	soft_attack_en	W/R	软攻击使能，置1后不能清0，除非掉电；

11.4.6 SEN状态寄存器（SEN_STATE）

偏移地址：0x00CC 复位值：0x00000000

31:21	20	19	18	17	16
预留	ssc_intp	soft_intp	mesh_intp	xtal32k_intp	vol_glitch_intp
ro	ro	ro	ro	ro	ro

15	14	13	12	11:0
tl_intp	th_intp	vl_intp	vh_intp	exts_intp
ro	ro	ro	ro	ro

Bit	Name	W/R	Description
31:21	预留	--	预留位
20	ssc_intp	RO	SSC产生攻击； 0：无攻击； 1：有攻击。
19	soft_intp	RO	软件置位攻击中断状态位； 0：无攻击； 1：有攻击。
18	mesh_intp	RO	Mesh攻击中断状态位； 0：无攻击； 1：有攻击。
17	xtal32k_intp	RO	32K时钟频率检测中断状态位； 0：无攻击； 1：有攻击。
16	vol_glitch_intp	RO	电压毛刺检测中断状态位； 0：无攻击； 1：有攻击。
15	tl_intp	RO	低温攻击中断状态位 0：无攻击； 1：有攻击。
14	th_intp	RO	高温攻击中断状态位 0：无攻击； 1：有攻击。
13	vl_intp	RO	低电压攻击中断状态位 0：无攻击； 1：有攻击。
12	vh_intp	RO	高电压攻击中断状态位 0：无攻击； 1：有攻击。

11:0	exts_intp	RO	外部传感器中断状态； 0：无攻击； 1：有攻击。 exts_intp[0]对应ext_s0，exts_intp[1]对应exts_s1，依次类推。 当为动态传感器时，只有传感器的输入端口对应bit会置1，如动态传感器1，exts_intp[3]与其对应，动态传感器2，exts_intp[5]与其对应。 对该寄存器进行写操作，清除所有中断；

11.4.7 SEN模拟传感器使能（SEN_ANA_EN）

偏移地址：0x00D4　　复位值：0x000000F0

31–16
预留
ro

15–8	7–4	3–1	0
预留	attack_type	预留	soft_attack
ro	rw	ro	wc

Bit	Name	W/R	Description
31:8	预留	--	预留位
7:4	attack_type	W/R	用于保存软件的攻击类型
3:1	预留	--	预留位
0	soft_attack	WC	软攻击位，软件置1后，产生攻击； 软件置1，硬件清0；

11.4.8 SEN模拟传感器门限（SEN_ANA_THR）

偏移地址：0x00D8　　复位值：0x80000000

31	30–16
soft_attack_lock	预留
rw	ro

15–0
预留
ro

Bit	Name	W/R	Description
31	soft_attack_lock	RW	软攻击解锁： 0：软攻击解锁，对soft_attack置1后可产生攻击； 1：软攻击上锁，无法对soft_attack置1。 解锁后需立即对soft_attack置1，对BPU_SEN寄存器的写操作将会使soft_attack_lock上锁。
30:0	预留	--	预留位

11.4.9 SEN攻击次数计数（SEN_ATTACK_CNT）

偏移地址：0x00DC 复位值：0x0000000F

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
预留															
ro															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
预留												attack_cnt			
ro												rw			

Bit	Name	W/R	Description
31:4	预留	--	预留位
3:0	attack_cnt	RW	仅用于软件记录攻击次数

11.4.10电压毛刺检测（SEN_VG_DETECT）

偏移地址：0x00E4 复位值：0x00000000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
预留															
ro															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
预留													ssc_dly_bypass	ssc_dly_sel	
ro													rw	rw	

Bit	Name	W/R	Description
31:3	预留	--	预留位
2	ssc_dly_bypass	W/R	0：开启电压毛刺检测； 1：关闭电压毛刺检测。
1:0	ssc_dly_sel	W/R	用于选择多大的毛刺宽度才能触发攻击 00：250us； 01：1ms 10：4ms； 11：16ms

11.4.11伪随机数初始值（SEN_RNG_INI）

偏移地址：0x00E8 复位值：0x00000000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
rng_ini															
ro															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
rng_ini															
ro															

Bit	Name	W/R	Description
31:0	rng_ini	RO	RNG随机值初始化寄存器

11.4.12Tamper使能控制（SEN_EXTx_EN）（x=0…7）

偏移地址：0x0104 – 0x0120 复位值：0x000000AA

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
预留															
ro															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
预留								extsx_en							
ro								rw							

Bit	Name	W/R	Description
31:8	预留	--	预留位
7:0	extsx_en	W/R	不等于55时，外部Tamper x使能。x=0…7

11.4.13Sensor高压使能（SEN_VH_EN）

偏移地址：0x0134　　复位值：0x000000AA

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
预留															
ro															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
预留								vh_en							
ro								rw							

Bit	Name	W/R	Description
31:8	预留	RO	预留位
7:0	vh_en	W/R	不等于55时，高压Sennsor使能。

11.4.14Sensor低压使能（SEN_VL_EN）

偏移地址：0x0138　　复位值：0x000000AA

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
预留															
ro															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
预留								vl_en							
ro								rw							

Bit	Name	W/R	Description
31:8	预留	--	预留位
7:0	vl_en	W/R	不等于55时，低压Sennsor使能。

11.4.15Sensor高温使能（SEN_TH_EN）

偏移地址：0x013C　　复位值：0x000000AA

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
预留															
ro															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
预留								th_en							
ro								rw							

Bit	Name	W/R	Description
31:8	预留	--	预留位
7:0	th_en	W/R	不等于55时，高温Sennsor使能。

11.4.16Sensor低温使能（SEN_TL_EN）

偏移地址：0x0140 复位值：0x000000AA

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
预留															
ro															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
预留								tl_en							
ro								rw							

Bit	Name	W/R	Description
31:8	预留	--	预留位
7:0	tl_en	W/R	不等于55时，低温Sennsor使能。

11.4.1732K频率传感器使能（SEN_XTAL32_EN）

偏移地址：0x0144 复位值：0x000000AA

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
预留															
ro															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
预留								xtal32k_en							
ro								rw							

Bit	Name	W/R	Description
31:8	预留	--	预留位
7:0	xtal32k_en	W/R	不等于55时，XTAL32K Sennsor使能。

11.4.18Active shielding使能（SEN_MESH_EN）

偏移地址：0x0148 复位值：0x80000055

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
pd_mesh	预留														
rw	ro														

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
预留								mesh_en							
ro								rw							

Bit	Name	W/R	Description
31	pd_mesh	W/R	MESH掉电
30:8	预留	--	预留位
7:0	mesh_en	W/R	不等于55时，active shielding使能。

11.4.19电压毛刺检测使能（SEN_VOLGLITCH_EN）

偏移地址：0x014C 复位值：0x000000AA

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
预留															
ro															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
预留								vol_glitch_en							
ro								rw							

Bit	Name	W/R	Description

31:8	预留	--	预留位
7:0	vol_glitch_en	W/R	不等于55时，电压毛刺检测使能。

11.4.20外部传感器启动位（SEN_EXTS_START）

偏移地址：0x0150　　复位值：0x000000AA

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
exts_state	预留														
ro	ro														
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
预留								exts_start							
ro								rw							

Bit	Name	W/R	Description
31	exts_state	W/R	外部传感器工作状态位， 1：外部传感器工作中 0：外部传感器停止中 exts_start启动/关闭Tamper后，需等待3个32K时钟，该状态位才会更新 软件只有当查询到exts_state为0时，才可以配置外部Tamper的相关寄存器
30:8	预留	--	预留位
7:0	exts_start	W/R	寄存器值： 非0x55：启动外部传感器； 0x55：关闭外部传感器。

11.4.21EXTS锁定（SEN_EXTS_LOCK）

偏移地址：0x0154　　复位值：0x00000000

31	30	29	28	27	26	25	24
预留							
ro							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
预留							
ro							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
预留							
ro							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
mesh_lock	xtal32k_lock	vol_glitch_lock	tl_lcok	th_lock	vl_lock	vh_lock	exts_lock
rw	rw	rw	rw	rw	rw	rw	rw

Bit	Name	W/R	Description
31:8	exts_state	RO	预留位，读取值为0。
7	mesh_lock	W/R	Mesh使能锁定，锁定后相关使能寄存器无法写入 1：锁定 0：不锁定
6	xtal32k_lock	W/R	32K时钟频率使能锁定，锁定后相关使能寄存器无法写入

			1：锁定 0：不锁定
5	vol_glitch_lock	W/R	电压毛刺使能锁定，锁定后相关使能寄存器无法写入 1：锁定 0：不锁定
4	tl_lcok	W/R	低温传感器使能锁定，锁定后相关使能寄存器无法进行写入 1：锁定 0：不锁定
3	th_lock	W/R	高温传感器使能锁定，锁定后相关使能寄存器无法进行写入 1：锁定 0：不锁定
2	vl_lock	W/R	低电压传感器使能锁定，锁定后相关使能寄存器无法进行写入 1：锁定 0：不锁定
1	vh_lock	W/R	高电压攻击使能锁定，锁定后相关使能寄存器无法进行写入 1：锁定 0：不锁定
0	exts_lock	W/R	外部传感器使能锁定，锁定后相关使能寄存器无法进行写入 1：锁定 0：不锁定 备注：锁定寄存器均为单向配置，置1后无法清0，只能通过BPK_RR或VBAT掉电复位清0

11.4.22模拟控制寄存器（SEN_ANA0）

偏移地址：0x0158 复位值：0x03500220

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
pk_ctrl	xtal_pd	xtal_current_sel			预留										
wc		rw			ro										

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
预留															
ro															

Bit	Name	W/R	Description
31	pk_ctrl	WC	控制EN_LDO5V输出电平 置1 EN_LDO5V电平翻转 软件置1，硬件清0
30	xtal_pd	RW	XTAL 32.768K powerdown控制；0：powerdown，切换至内部RC；1：工作
29:27	xtal_current_sel	RW	XTAL 32.768K晶振电流选择
26:0	保留	--	保留，用户不能随机修改该寄存器的值

11.4.23攻击上拉清除寄存器（SEN_ATTCLR）

偏移地址：0x0160 复位值：0x00000000

31	30	29	28	27	26	25	24
预留							
ro							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
预留							
ro							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
预留							
ro							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
attclr							
rw							

Bit	Name	W/R	Description
31:8	rsvd	RO	保留，用户不能随机修改该寄存器的值
7:0	attclr	RW	对相应位写1，清除对应Sensor的攻击后上拉 0对应Sensor0； 1对应Sensor1； 2对应Sensor2，以此类推

11.4.24脉冲上拉Mask寄存器（SEN_PULL_MASK）

偏移地址：0x0174 复位值：0xFF0000FF

31	30	29	28	27	26	25	24
pull_mask							
rw							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
预留							
ro							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
预留							
ro							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
预留							
ro							

Bit	Name	W/R	Description
31:24	pull_mask	RW	分别对应8路Tamper的脉冲上拉使能 0：关闭对应Tamper的脉冲上拉 1：开启对应Tamper的脉冲上拉
23:0	rsvd	RO	预留

12 看门狗（WDT）

12.1 看门狗外设时钟

看门狗外设时钟由PCLK提供，即看门狗外设时钟频率等于PLCK时钟频率。

12.2 计数器（Counter）

看门狗计数器（DWT_CCVR：Watchdog Timer Current Counter Value Register）为递减计数器，即计数器值由预设值递减直至数值为0。当计数器计数到0时，看门狗根据设定模式产生系统复位或中断。

工作在中断模式下的看门狗，在看门狗计数器第1次计数到0时，会产生看门狗中断，并重置看门狗计数器到预设值，但不会产生系统复位。此后看门狗计数器会进入下一轮递减计数，用户必须在此次计数过程中进行喂狗或清中断处理操作，否则在此次计数至0后，系统发生复位。

用户可以在发生复位前任意时刻，通过对计数器重置寄存器（WDT_CRR：Counter Restart Register）写0x76，来重置计数器为预设值。完成“喂狗”操作。

12.3 计数器预设值（Timeout Period Values）

看门狗计数器预设值由WDT_RLD（Watchdog Timer Reload Value Register）保存，用户可以通过该寄存器设定看门狗计数器超时时间。对WDT_CRR寄存器写0x76将对DWT_CCVR寄存器的置重置为此预设值，完成“喂狗”操作。

12.4 启用看门狗（WatchDog Enable）

看门狗开启由看门狗控制寄存器（WDT_CR：Watchdog Timer Control Register）控制，当WDT_CR[0] = 1时看门狗开启。看门狗使能开启后将无法关闭，可通过复位看门狗模块来关闭看门狗。

12.5 系统复位/中断（System Resets）

看门狗包含2中模式：
WDT_CR[1] = 0：系统复位模式
WDT_CR[1] = 1：中断模式
系统复位模式：
看门狗计数器计数到0后，系统立即产生复位。
中断模式：
在看门狗计数器第1次计数到0时，会产生看门狗中断（中断源为不可屏蔽中断NMI），并重置看门狗计数器到预设值，但不会产生系统复位。此后看门狗计数器会进入下1轮递减计数，用户必须在此次计数过程中进行喂狗或清中断处理操作，否则在此次计数至0后，系统发生复位。

运行在中断模式中的看门狗，除了使用普通喂狗方式（重置看门狗计数器）外，还可以通过清除看门狗中断标记完成喂狗。

看门狗中断可以通过以下两种方式清除：

1、重置看门狗计数器（喂狗）
对WDT_CRR寄存器写0x76后，硬件自动将WDT_RLD寄存器的值加载到DWT_CCVR寄存器，完成“喂狗”操作。

2、读看门狗中断清除寄存器（WDT_EOI）
对WDT_EOI寄存器进行读操作清除看门狗中断标记。

12.6 寄存器描述

12.6.1 地址映射表

WDT基地址列表

地址范围	基地址	外设	总线
0x4001_C000-0x4001_CFFF	0x4001_C000	WDT	APB0

表 12-1 WDT寄存器表

偏移地址	寄存器名称	宽度（bit）	复位值
0x00	WDT_CR	32	0x00000000
0x04	预留	32	0x00000000
0x08	WDT_CCVR	32	0x0000FFFF
0x0C	WDT_CCR	32	0x00000000
0x10	WDT_STAT	32	0x00000000
0x14	WDT_EOI	32	0x00000000
0x18	预留	32	0x00000000
0x1C	WDT_RLD	32	0xFFFFFFFF

12.6.2 看门狗控制寄存器（WDT_CR）

- **Name:** Control Register
- **Size:** 32bits
- **Address Offset:** 0x00
- **Read/write access:** read/write

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
预留															
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
预留														RMOD	WDT_EN

Bit	Name	R/W	Description
31:2	预留	--	读操作返回 0
1	RMOD	R/W	Response mode 选择看门狗超时应答模式 0 = 产生系统复位 1 = 第 1 次看门狗超时产生系统中断，第 2 次看门狗超时产生前未清除中断，则发生系统复位。 复位值：0x00
0	WDT_EN	R/W	WDT enable 看门狗使能操作位。看门狗使能开启后将无法关闭，系统复位不影响看门狗使能。 0 = 看门狗关闭 1 = 看门狗开启 复位值：0x00

12.6.3 看门狗计数器（WDT_CCVR）

- **Name:** Current Counter Value Register
- **Size:32bits**
- **Address Offset:** 0x08
- **Read/write access:** read

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
WDT_CCVR_H															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WDT_CCVR_L															

Bit	Name	R/W	Description
31:0	WDT_CCVR	R	对该寄存器读操作，读取的值为读取时刻对应的计数值。 复位值：0xFFFF

12.6.4 看门狗计数器重置寄存器（WDT_CRR）

- **Name:** Counter Restart Register
- **Size:** 32 bits
- **Address Offset:** 0x0c
- **Read/write access:** write

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
预留															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
预留								WDT_CRR							

Bit	Name	R/W	Description
31:8	预留	-	读操作返回 0
7:0	WDT_CRR	W	该寄存器用于重置看门狗计数器值，为防止意外操作，写入值必须是0x76。对该寄存器读返回值为0 复位值：0x00

12.6.5 看门狗中断状态寄存器（WDT_STAT）

- **Name:** Interrupt Status Register
- **Size:** 32 bit
- **Address Offset:** 0x10
- **Read/write access:** read

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
预留															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
预留															STAT

Bit	Name	R/W	Description
31:1	预留	-	读操作返回 0
0	STAT	R	Interrupt Status

			该位用于表示看门狗的中断状态 1 = 看门狗中断产生 0 = 看门狗中断未产生 复位值：0x00

12.6.6 看门狗中断清除寄存器（WDT_EOI）

- **Name:** Interrupt Clear Register
- **Size:** 32 bit
- **Address Offset:** 0x14
- **Read/write access:** read

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
预留															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
预留															EOI

Bit	Name	R/W	Description
31:1	预留	--	读操作返回 0
0	EOI	R	Clears the watchdog interrupt 清除看门狗中断，并重置看门狗计数器（WDT_CCVR）。 复位值：0x00

12.6.7 看门狗预设值寄存器（WDT_RLD）

- **Name:** Reload Value Register
- **Size:** *32bits*
- **Address Offset:** 0x1C
- **Read/write access:** read/write

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
WDT_RLD_H															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WDT_RLD_L															

Bit	Name	R/W	Description
31:0	WDT_RLD	R/W	存放看门狗计数器预设值 复位值：0xFFFFFFFF

13 定时器（TIMER）

13.1 定时器简介

- 1 个 Timer 单元，包含 6 个独立定时器（Timer0，Timer1，Timer2，Timer3，Timer4，Timer5），共 6 个定时器。
- 6 个定时器中断源独立，每个定时器单独占 1 个中断源
- 定时器采用向下计数方式
- 每个 Timer 单元定时器都支持 PWM 模式
- 使用 PCLK 时钟频率作为定时器计时钟源
- PWM 单次触发(one shot)功能

13.2 定时器外设时钟

定时器外设时钟由PCLK提供，即定时器时钟频率等于PCLK外设时钟频率

13.3 通用定时器

13.3.1 通用定时器的两种模式

在自由运行（free-running）和用户定义（user-defined）模式下，当定时器使能后计数值由TimerNLoadCount寄存器载入。

根据选择的模式选择载入值：

用户定义模式（user-defined）：

定时器计数值载入TimerNLoadCount寄存器设定值。使用用户模式可以产生固定时间的定时器中断。

自由运行模式（free-runing）：

定时器计数值会载入其允许的最大值，即0xFFFFFFFF。在定时器产生中断（定时器计数器计数到0）前用户可以再编程或禁止定时器中断。使用该模式，定时器只产生1次中断。中断产生后计数值重置为0xFFFFFFFF并向下计数，但不会再产生中断。

13.3.2 中断处理

定时器产生中断后，用户可以通过对TimerNEOI或TimersEOI进行读操作清除定时器中断状态。

TimerNEOI：只清除对应定时器中断源上的中断状态。

TimersEOI：清除该定时器单元中的定时器中断状态。

13.4 PWM模式

Timer单元的6个独立定时器均可编程产生PWM信号。

当用户设定TimerNControlReg中PWM比特位为“1”后，定时器进入PWM工作模式。此时PWM由TimerNLoadCount2和TimerNLoadCount寄存器分别控制高电平及低电平周期翻转输出。

13.4.1 PWM工作模式

设定TimerNControlReg中PWM位为“1”，定时器使能后工作在PWM模式。

13.4.2 PWM周期及占空比设定

PWM信号频率及占空比可通过以下方式进行配置：

- Width of HIGH period = (TimerNLoadCount2 + 1) * PCLK_ Period
- Width of LOW period = (TimerNLoadCount + 1) * PCLK_ Period

13.5 寄存器描述

13.5.1 地址映射表

TIMER基地址列表

地址范围	基地址	外设	总线
0x4001_3000-0x4001_3FFF	0x4001_3000	Timer	APB0

表 13-1 TIMER寄存器表

偏移地址	寄存器名称	宽度（bit）	复位值
0x00	Timer0LoadCount	32	0x00000000
0x04	Timer0CurrentValue	32	0xFFFFFFFF
0x08	Timer0ControlReg	32	0x00000000
0x0C	Timer0EOI	32	0x00000000
0x10	Timer0IntStatus	32	0x00000000
0x14	Timer1LoadCount	32	0x00000060
0x18	Timer1CurrentValue	32	0xFFFFFFFF
0x1C	Timer1ControlReg	32	0x00000000
0x20	Timer1EOI	32	0x00000000
0x24	Timer1IntStatus	32	0x00000000
0x28	Timer2LoadCount	32	0x00000060
0x2C	Timer2CurrentValue	32	0xFFFFFFFF
0x30	Timer2ControlReg	32	0x00000000
0x34	Timer2EOI	32	0x00000000
0x38	Timer2IntStatus	32	0x00000000
0x3C	Timer3LoadCount	32	0x00000060
0x40	Timer3CurrentValue	32	0xFFFFFFFF
0x44	Timer3ControlReg	32	0x00000000
0x48	Timer3EOI	32	0x00000000
0x4C	Timer3IntStatus	32	0x00000000
0x50	Timer4LoadCount	32	0x00000060
0x54	Timer4CurrentValue	32	0xFFFFFFFF
0x58	Timer4ControlReg	32	0x00000000
0x5C	Timer4EOI	32	0x00000000
0x60	Timer4IntStatus	32	0x00000000
0x64	Timer5LoadCount	32	0x00000060
0x68	Timer5CurrentValue	32	0xFFFFFFFF
0x6C	Timer5ControlReg	32	0x00000000
0x70	Timer5EOI	32	0x00000000
0x74	Timer5IntStatus	32	0x00000000
0x78 - 0x9F	预留 x 120	32 x 120	0x00000000
0xA0	TimersIntStatus	32	0x00000000
0xA4	TimersEOI	32	0x00000000
0xA8	TimersRawIntStatus	32	0x00000000
0xAC	预留	32	0x00000000
0xB0	Timer0LoadCount2	32	0x00000000
0xB4	Timer1LoadCount2	32	0x00000000
0xB8	Timer2LoadCount2	32	0x00000000
0xBC	Timer3LoadCount2	32	0x00000000

0xC0	Timer4LoadCount2	32	0x00000000
0xC4	Timer5LoadCount2	32	0x00000000

13.5.2 自动重载计数器（TimerNLoadCount）（N=0…5）

- **Name: TimerN Load Count Register**
- **Size:** 32 bits
- **Address Offset:**
  for N = 1, 0x00
  for N = 2, 0x14
  for N = 3, 0x28
  for N = 4, 0x3C
  for N = 5, 0x50
  for N = 6, 0x64
- **Read/write access:** read/write

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Load Count															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Load Count															

Bit	Name	R/W	Description
31:0	Load Count	R/W	该值被自动加载到 TimerN 中计数。

13.5.3 自动重载计数器2（TimerNLoadCount2）（N=0…5）

- **Name: TimerN Load Count Register 2**
- **Size:** 32 bits
- **Address Offset:**
  for N = 1, 0xb0
  for N = 2, 0xb4
  for N = 3, 0xb8
  for N = 4, 0xbc
  for N = 5, 0xc0
  for N = 6, 0xc4
- **Read/write access:** read/write

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Load Count2															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Load Count2															

Bit	Name	R/W	Description
31:0	Load Count2	R/W	当定时器工作在PWM模式中时，该值被自动加载到TimerN中计数。当该值被加载到 TimerN 中， 在此计数期间 PWM 输出保持高电平。

13.5.4 当前计数器值（TimerNCurrentValue）（N=0…5）

- **Name: TimerN Current Value Register**
- **Size:** 32 bits
- **Address Offset:**

for N = 1, 0x04
for N = 2, 0x18
for N = 3, 0x2c
for N = 4, 0x40
for N = 5, 0x54
for N = 6, 0x68

- **Read/write access:** read only

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CurrentValue															
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
CurrentValue															

Bit	Name	R/W	Description
31:0	CurrentValue	R	TimerN 当前计数值。

13.5.5 控制寄存器（TimerNControlReg）（N=0…5）

- **Name: TimerN Control Register**
- **Size:** 32 bits
- **Address Offset:**

for N = 1, 0x08
for N = 2, 0x1C
for N = 3, 0x30
for N = 4, 0x44
for N = 5, 0x58
for N = 6, 0x6C

- **Read/write access:** read/write

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
预留															
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
										Tim_Reload	PWM_Oneshot	PWM	IntMask	Mode	Enable

Bit	Name	R/W	Description
31:6	预留	-	保留位
5	Tim_Reload	W	PWM Oneshot 重新加载 TimerNLoadCount2 寄存器值
4	PWM_Oneshot	R/W	PWM Oneshot 使能位： 0-PWM Oneshot 模块关闭 1-PWM Oneshot 模式打开
3	PWM	R/W	PWM 使能位： 0-PWM 模式关闭 1-PWM 模式打开
2	IntMask	R/W	中断屏蔽位： 0-中断打开 1-中断屏蔽
1	Mode	R/W	通用定时器工作模式： 0-自由运行模式（free-runing）； 1-用户定义模式（user-defined）；
0	Enable	R/W	定时器使能位： 0-关闭 1-打开

13.5.6 中断清除寄存器（TimerNEOI）（N=0…5）

- **Name: TimerN End-of-Interrupt Register**
- **Size:** 32 bits
- **Address Offset:**
  - for N = 1, 0x0C
  - for N = 2, 0x20
  - for N = 3, 0x34
  - for N = 4, 0x48
  - for N = 5, 0x5c
  - for N = 6, 0x70
- **Read/write access:** read only

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
预留															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
预留															TimerNEOI

Bit	Name	R/W	Description
31:1	预留	-	
0	TimerNEOI	RC	对该位读操作清除定时器中断状态。返回值为 0

13.5.7 中断状态寄存器（TimerNIntStatus）（N=0…5）

- **Name: TimerN Interrupt Status Register**
- **Size:** 32bits
- **Address Offset:**
  - for N = 1, 0x10
  - for N = 2, 0x24
  - for N = 3, 0x38
  - for N = 4, 0x4c
  - for N = 5, 0x60
  - for N = 6, 0x74
- **Read/write access:** read only

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
预留															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
预留															IntStatus

Bit	Name	R/W	Description
31:1	预留	-	
0	IntStatus	R	定时器中断标志位，定时器产生中断该位为“1”。

13.5.8 全局中断清除寄存器（TimersEOI）

- **Name: Timers End-of-Interrupt Register**
- **Size:** 32 bits
- **Address Offset: 0xa4**
- **Read/write access:** read only

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
预留															
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
预留															TimersEOI

Bit	Name	R/W	Description
31:1	预留	-	
0	TimersEOI	RC	对该位读操作清除定时器单元中断状态

13.5.9 全局原中断状态寄存器（TimersRawIntStatus）

- **Name: Timers Interrupt Status Register**
- **Size:**32 bits
- **Address Offset: 0xa8**
- **Read/write access:** read only

该寄存器中断标志值不受寄存器TimerNControlReg中IntMask影响。

寄存器TimersIntStatus的值是寄存器TimersRawIntStatus经IntMask后的值。当TimersIntStatus有效位置“1”，Timer向CPU发出中断请求。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
预留															
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
预留															RawIntStatus

Bit	Name	R/W	Description
31:1	预留	-	
0	RawIntStatus	R	定时器单元原中断标志位，定时器产生中断该位为“1”。

14 通用异步收发器（UART）

14.1 UART简介

通用异步收发器(UART)提供了一种灵活的方法与使用工业标准NRZ异步串行数据格式的外部设备之间进行全双工数据交换。UART利用波特率发生器提供宽范围的波特率选择。

通用异步收发器(UART)支持单向通信、双工通信和IrDA(红外数据组织)SIR ENDEC规范，以及调制解调器(CTS/RTS)操作。

与DMA配合使用，可以实现高速数据通信。

14.2 串行红外协议（IrDA 1.0 SIR Protocol）

14.2.1 串行红外协议介绍

红外线1.0串行红外模式支持红外设备数据的双向数据传输。IrDA 1.0 SIR模式最大波特率为115.2K。

数据格式类似于标准串口的数据格式，每个数据字节都包含以下部分：

1、1个bit为作为起始位
2、紧接的8bit为数据位
3、至少1个bit的结束位

传输数据位固定。不支持校验位，并且在这此模式中只有1个停止位。使用LCR寄存器设置数据的比特数和使能校验位无效。

IrDA脉冲定义：

1、产生脉冲信号时表示逻辑0；
2、无脉冲信号时表示逻辑1；

IrDA脉冲宽度只有普通串口脉冲宽度的3/16；字节传输的起始位由1个脉冲表示。但是在IrDA接收器上的由于光电二极管被红外线激励问题，会导致UART接收的数据与实际发送数据相位的翻转，即高电平变为低电平。该晶体管电平得翻转由UART外设输入端转化为UART当前需要的电平。

14.2.2 SIR模式使能

UART外设IrDA 1.0 SIR模式，通过模式控制寄存器MCR[6] = 0使能打开。

14.2.3 SIR模式操作特点

IrDA SIR模式，数据的传输只能是半双工模式。这个主要是由于IrDA SIR物理层的指定在发送和接收之间必须要由10ms的延时。该10ms的延时由软件控制。

14.3 接收/发送FIFO

14.3.1 接收/发送FIFO介绍

UART外设可配置使用FIFO进行收发数据。每个UART外设均包括16bytes的独立的接收和发送FIFO。

14.3.2 接收/发送FIFO

当用户使能FIFO后，CPU写THR寄存器的数据被保存到发送FIFO中，UART外设接收到的数据被保存到接收FIFO中。用户可以通过UART状态寄存器（USR）等相关寄存器获取FIFO中有效数据数量或FIFO状态。也可以配置UART中断，设定在特定条件下向CPU产生中断请求后处理FIFO中数据。

14.3.3 接收/发送FIFO中断使用

触发阈值配置：

用户可使用FIFO控制寄存器（FCR）或其对应的影子寄存器配置接收/发送FIFO数据中断触发的阈值。当接收/发送FIFO中的数据满足设定阈值后会触发对应使能的FIFO中断。

接收FIFO中断：

FIFO模式启用，ERBFI中断使能，当接收FIFO中接收的数据量满足设定阈值后触发“接收数据有效中断”。

发送FIFO中断：

FIFO模式启用，ETBEI中断时能，并且可编程THRE中断模式使能(IER[7] = 1)，当发送FIFO中剩余的数据量满足设定阈值后触发“发送寄存器空中断”。在数据超时未取的情况下触发“字符超时中断”。

14.3.4 接收/发送FIFO访问模式

UART外设提供FIFO访问模式，该模式用于程序调试。在FIFO访问模式中，CPU可对接收FIFO进行写操作，对发送FIFO进行读操作。

进入访问模式：

FAR[0] = 1，FIFO访问模式（FIFO Access mode）使能打开，使能打开后发送和接收FIFO被硬件清空。

发送FIFO测试：

在FIFO访问模式下，写入到发送FIFO中的数据不会被移位发送，而是一直预留在发送FIFO。用户可以通过对TFR寄存器进行读取FIFO中数据，用于测试数据的正确性。

接收FIFO测试：

在FIFO访问模式下，用户通过对RFW(Receive FIFO Write) 寄存器写操作向接收FIFO中写入数据，其中RFW[9]用于测试帧错误的产生，RFW[8]用于测试校验错误产生。数据可以由接收FIFO正常的回读。由于在FIFO访问模式下正常的操作被禁止，所以数据必人为写入到接收FIFO，无法有外部接收。

14.4 UART外设时钟

UART外设时钟由内部PCLK提供，即UART外设时钟频率等于PCLK时钟频率。

14.5 中断（Interrupt）

UART外设产生中后，用户可以通过IIR寄存器获取中断类型。

UART外设可产生中断类型如下：

- Modem 状态中断
- 发送寄存器空中断
- 接收数据有效中断
- Line 状态中断

- UART 忙中断
- 字符超时中断

14.6 可编程THRE中断（Programmable THRE Interrupt）

UART外设可以通过编程THRE中断模式来实现。

THRE中断模式设定关系如下：

THRE中断	可编程THRE中断模式使能(IER[7])	FIFO使能	THRE中断使能(IER[1])
无中断	-	-	×
THR寄存器为空时产生中断	-	×	√
THR和发送FIFO均为空时产生中断	×	√	√
发送FIFO数据到达或低于设定阈值时产生中断	√	√	√

发送FIFO可设置的空阈值（FCR[5:4]）如下：

- ■ empty
- ■ 2chars
- ■ ¼ Full
- ■ ½ Full

14.7 DMA支持

UART外设使用DMA功能可以有的效减少系统中断，提高数据传输效率。每个UART外设可以使用2个DMA通道，分别用来接收和发送数据。

UART外设使用DMA功能时可以选择是否使用UART FIFO。在不使用FIFO情况下，UART在每次发完成或接收到数据后向DMA发出请求。使用FIFO情况下，UART在每次接收/发送FIFO中数据量满足触发阈值后想DMA发出请求。

DMA模式1中DMA可以连续的搬运数据直到接收FIFO为空或发送FIFO满为止。该模式下必须使用UART FIFO，即使能UART FIFO（FCR[0] = 1）。

1、UART外设的DMA发送请求：

- 在以下情况下请求信号有效：
  a) FIFO使能打开（FCR[0] = 1）且THRE中断模式使能关闭（IRE[7] = 0），发送FIFO为空。
  b) FIFO使能打开（FCR[0] = 1）且THRE中断模式使能打开（IRE[7] = 1），发送FIFO数据量达到或低于设定的阈值。
- 在以下情况下请求信号无效：
  FIFO使能打开（FCR[0] = 1），DMA通道连续写发送FIFO，直到发送FIFO为满时，请求信号无效。

2、UART外设的DMA接收请求：

- 在以下情况下请求信号有效：
  a) FIFO使能打开（FCR[0] = 1），接收FIFO中数据达到或高于设定的阈值。
  b) FIFO使能打开（FCR[0] = 1），FIFO中发生字节数据超时（character timeout），不需使能ERBFI（IER[0] = 1）中断。
- 在以下情况下请求信号无效：
  FIFO使能打开（FCR[0] = 1），DMA通道连续读接收FIFO，直到接收FIFO为空是，请求信号无效。

14.8 寄存器描述

14.8.1 地址映射表

UARTx（x=0…1）基地址列表

地址范围	基地址	外设	总线
0x4001_6000-0x4001_6FFF	0x4001_6000	UART0	APB0
0x4001_7000-0x4001_7FFF	0x4001_7000	UART1	
0x4001_5000-0x4001_5FFF	0x4001_5000	UART2	

表 14-1 UART寄存器表

偏移地址	寄存器名称	宽度（bit）	复位值
0x00	DLL/THR/RBR	32	0x00000000
0x04	DLH/IER	32	0x00000000
0x08	IIR/FCR	32	IIR = 0x00000001 FCR = 0x00000000
0x0C	LCR	32	0x00000000
0x10	MCR	32	0x00000000
0x14	LSR	32	0x00000060
0x18	MSR	32	0x00000000
0x1C	SCR	32	0x00000000
0x20	预留	32	0x00000000
0x24	预留	32	0x00000000
0x28	预留	32	0x00000000
0x2C	预留	32	0x00000000
0x30	SRBR/STHR	32	0x00000000
0x34	SRBR/STHR	32	0x00000000
0x38	SRBR/STHR	32	0x00000000
0x3C	SRBR/STHR	32	0x00000000
0x40	SRBR/STHR	32	0x00000000
0x44	SRBR/STHR	32	0x00000000
0x48	SRBR/STHR	32	0x00000000
0x4C	SRBR/STHR	32	0x00000000
0x50	SRBR/STHR	32	0x00000000
0x54	SRBR/STHR	32	0x00000000
0x58	SRBR/STHR	32	0x00000000
0x5C	SRBR/STHR	32	0x00000000
0x60	SRBR/STHR	32	0x00000000
0x64	SRBR/STHR	32	0x00000000
0x68	SRBR/STHR	32	0x00000000
0x6C	SRBR/STHR	32	0x00000000
0x70	FAR	32	0x00000000
0x74	TFR	32	0x00000000
0x78	RFW	32	0x00000000
0x7C	USR	32	0x00000006
0x80	TFL	32	0x00000000
0x84	RFL	32	0x00000000
0x88	SRR	32	0x00000000
0x8C	SRTS	32	0x00000000
0x90	SBCR	32	0x00000000
0x94	SDMAM	32	0x00000000
0x98	SFE	32	0x00000000
0x9C	SRT	32	0x00000000
0xA0	STET	32	0x00000000
0xA4	HTX	32	0x00000000

0xA8	DMASA	32	0x00000000
0xAC~0xFC	预留	32	0x00000000

14.8.2 接收缓存寄存器（RBR）

- **Name:** Receive Buffer Register
- **Size:** 32 bits
- **Address Offset:** 0x00
- **Read/write access:** read-only

RBR寄存器只有当DLAB比特位清0后才可以访问。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
预留															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
预留								Receive Buffer Register							

Bit	Name	R/W	Description
7:0	Receive Buffer Register	R	接收数据寄存器 UART外设在串口或红外模式下用于接收数据。 UART外设成功接收数据后，LSR寄存器中DR位置“1”。 在非FIFO模式下，未及时读取的数据会被下次接收的数据覆盖，并产生溢出错误标志。 在FIFO模式下，新接收数据被保存在FIFO头部。若FIFO以满，后续接收的数据会被丢弃，并产生溢出错误标志。 复位值：0x00

14.8.3 发送保持寄存器（THR）

- **Name:** Transmit Holding Register
- **Size:** 32 bits
- **Address Offset:** 0x00
- **Read/write access:** write-only

THR寄存器只有当DLAB比特位清0后才可以访问。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
预留															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
预留								Transmit Holding Register							

Bit	Name	R/W	Description
7:0	Transmit Holding Register	W	发送数据寄存器 UART外设在串口或红外模式下用于发送数据。 在非FIFO模式，数据发送保持位（THRE）置“1”，发送数据寄存器为空，数据发送保持位（THRE）清“0”，发送数据寄存器不为空。 在FIFO模式下，在FIFO未满的情况下可以连续写入数据，当FIFO已满继续写入的数据将丢失。 复位值：0x00

14.8.4 分频寄存器_高（DLH）

- **Name:** Divisor Latch High
- **Size:** 32 bits
- **Address Offset:** 0x04
- **Read/write access:** read/write

DLH寄存器只有在LCR寄存器的DLAB位置1并且UART不是忙状态（USRT bit0 为0）时才能够访问。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
预留															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
预留								Divisor Latch(High)							

Bit	Name	R/W	Description
7:0	Divisor Latch (High)	R/W	波特率分频寄存器高位。 波特率=输入时钟/(16*分频波特率寄存器值)，UART输入时钟即PCLK时钟周期 当波特率分频寄存器(DLL和DLH)的值设置为0，波特率时钟关闭，串口不可通信。 设定DLH寄存器后，UART经过8个波特率时钟后，进行数据接收或发送操作。 复位值：0x00

14.8.5 分频寄存器_低（DLL）

- **Name:** Divisor Latch Low
- **Size:** 32 bits
- **Address Offset:** 0x00
- **Read/write access:** read/write

DLL寄存器只有在LCR寄存器的DLAB位置1并且UART不是忙状态（USRT bit0 为0）时才能够访问。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
预留															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
预留								Divisor Latch(Low)							

Bit	Name	R/W	Description
7:0	Divisor Latch (Low)	R/W	波特率分频寄存器低位。 波特率=输入时钟/(16*分频波特率寄存器值)，UART输入时钟即PCLK时钟周期 当波特率分频寄存器(DLL和DLH)的值设置为0，波特率时钟关闭，串口不可通信。 设定DLH寄存器后，UART经过8个波特率时钟后，进行数据接收或发送操作。 复位值：0x00

14.8.6 中断使能寄存器（IER）

- **Name:** Interrupt Enable Register
- **Size:** 32 bits
- **Address Offset:** 0x04
- **Read/write access:** read/write

IER寄存器只有在LCR寄存器的DLAB位清0后才可以访问。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
预留															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
预留								PTIME	预留			EDSSI	ELSI	ETBEI	ERBFI

Bit	Name	R/W	Description
7	PTIME	R/W	Programmable THRE Interrupt Mode Enable 用于设定THRE中断模式，用来控制是否产生THRE中断 0 = disabled 1 = enabled 复位值：0x00
6:4	预留	--	预留
3	EDSSI	R/W	Enable Modem Status Interrupt 使能Modem状态中断，用来控制是否产生Modem状态中断。 中断优先级为4（优先相应高优先级中断）。 0 = disabled 1 = enabled 复位值：0x00
2	ELSI	R/W	Enable Receiver Line Status Interrupt 使能Line状态中断，用来控制是否产生Line状态中断。 中断优先级为1（优先相应高优先级中断）。 0 = disabled 1 = enabled 复位值：0x00
1	ETBEI	R/W	Enable Transmit Holding Register Empty Interrupt 使能发送寄存器空中断，用来控制是否产生发送寄存器空中断。 中断优先级为3（优先相应高优先级中断）。 0 = disabled 1 = enabled 复位值：0x00
0	ERBFI	R/W	Enable Received Data Available Interrupt 使能接收数据有效中断，用来控制是否产生接收数据有效中断和接收字节超时中断（在FIFO模式下或FIFO使能）。 中断优先级为2（优先相应高优先级中断）。。 0 = disabled 1 = enabled 复位值：0x00

14.8.7 中断标志寄存器（IIR）

- **Name:** Interrupt Identity Register
- **Size:** 32 bits
- **Address Offset:** 0x08
- **Read/write access:** read-only

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
预留															
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
预留								FIFOSE		预留		IID			

Bit	Name	R/W	Description
7:6	FIFOSE	R	FIFO State Enabled FIFO使能状态。 00 = disabled 11 = enabled 复位值：0x00
4:5	预留	-	预留
3:0	IID	R	Interrupt ID UART中断标志，对应中断类型（比特位表示） 0000 – Modem状态中断 0001 – 未发生中断 0010 – 发送寄存器空中断 0100 – 接收数据有效中断 0110 – Line状态中断 0111 – 忙中断 1100 – 字符超时中断（接收FIFO使能） 复位值：0x01

中断号	中断描述			
二进制表示	响应优先级	中断类型	中断源	中断清除操作
0001	-	无	无	
0110	1	Line状态中断	接收过载错误、校验错误、帧错误或接收到break中断	读Line中断状态寄存器（LSR）
0100	2	接收数据有效中断	①非FIFO模式中：接收到有效数据后 ②FIFO模式中：接收FIFO中的数据量达到设定阈值后	①非FIFO模式中：读取接收缓冲寄存器（RBR） ②FIFO模式中：读取接收FIFO中的数据，使接收FIFO中的数据量减少到触发阈值以下
1100	2	字节超时中断	在FIFO模式中，接收FIFO中有有效数据且在上个有效数据接收后的4单位时间内未接收到数据。 1单位时间：1个字节数据接收的用时	读取接收数据寄存器（RBR）
0010	3	发送寄存器空中断	①IRE寄存器PTIME位置0（IRE[7] = 0）：发送保持寄存器为空时 ②IRE寄存器PTIME位置1（IRE[7] = 1）：	如果中断源为①则给THR寄存器写数据或关闭发送空中断使能 如果中断源为②则写发送FIFO直到数据

			发送FIFO的数据低于设定的阈值时触发中断（IRE寄存器中PTIME置1）	高于设定的触发阈值或关闭发送空中断使能
0000	4	Modem状态中断	Modem状态寄存器相关位置1。 如果自动控制流使能，则CTS的改变不会引起中断	读Modem状态寄存器
0111	5	忙中断	UART处于忙状态时，尝试对LCR寄存器进行写操作	读USR寄存器

14.8.8 FIFO控制寄存器（FCR）

- **Name:** FIFO Control Register
- **Size:** 32 bits
- **Address Offset:** 0x08
- **Read/write access:** write-only

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
预留															
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
预留								RCVR		TET		DMAM	XFIFOR	RFIFOR	FIFOE

Bit	Name	R/W	Description
7:6	RCVR	W	Receiver Trigger 接收FIFO满触发的阈值设定。 当接收FIFO中的数据超过设定的阈值后，会触发接收数据有效中断。 在自动流控模式下，该阈值用于决定接收rts_n信号状态（在RTC_FCT使能关闭的状态下）。 在DMA模式下，FIFO中数据量触发阈值后UART发出DMA请求。 以下为支持的阈值类型。 00 – 1 character in the FIFO 01 – FIFO ¼ full 10 – FIFO ½ full 11 – FIFO 2 less than full 复位值：0x00
5:4	TET	W	TX Empty Trigger 发送FIFO空的阈值设定。 当PTIME使能打开(IER[7] = 1)，发送FIFO中数据等于或低于该阈值将触发THRE中断产生。 在DMA模式下，FIFO中数据量触发阈值后UART发出DMA请求。 以下为支持的阈值类型。 00 – FIFO empty 01 – 2 characters in the FIFO 10 – FIFO ¼ full 11 – FIFO ½ full 复位值：0x00

3	DMAM	W	DMA Mode 在额外DMA握手信号没有选择（DMA_EXTRE = NO）的情况下用于确定DMA发送请求和接收请求信号的模式。 0 = 不使用DMA 1 = 使用DMA 复位值：0x00
2	XFIFOR	W	XMIT FIFO Reset 发送FIFO复位。 复位发送FIFO相关状态信息并清空发送FIFO。复位会使DMA TX请求信号无效。 置“1”后，由硬件清“0”。 复位值：0x00
1	RFIFOR	W	RCVR FIFO Reset 接收FIFO复位。 复位接收FIFO相关状态信息并清空接收FIFO。复位会使DMA RX请求信号无效。 置“1”后，由硬件清“0”。 复位值：0x00
0	FIFOE	W	FIFO Enable 使能接收和发送FIFO。 使能操作会导致接收和发送FIFO复位。 复位值：0x00

14.8.9 Line控制寄存器（LCR）

- **Name:** Line Control Register
- **Size:** 32 bits
- **Address Offset:** 0x0C
- **Read/write access:** read/write

<table>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td colspan="16">预留</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="8">预留</td><td>DLAB</td><td>BC</td><td>SP</td><td>EPS</td><td>PEN</td><td>STOP</td><td colspan="2">DLS</td></tr>
</table>

Bit	Name	R/W	Description
31:8	预留	-	读操作返回 0
7	DLAB	R/W	Divisor Latch Access Bit DLL 和 DLH 读写操作使能位。 DLL/DLH 与其他寄存器地址复用，该为作为操作切换开关。 0-对相应复用寄存器操作 1-对 DLL/DLH 操作 置“1”后，需软件清“0”。 写操作，需要 USR[0]为 0，即 UART 外设不在忙状态。 复位值：0x00
6	BC	R/W	Break Control Bit 产生输出break信号到接收设备，该位置1，UART外设输出引脚会强行输出逻辑0信号。 在非回环模式中（MCR[4] = 0），UART模式（MCR[4] = 0），sout输出将强制拉低直。红外模式（MCR[4] = 1），sout持续输出脉冲信号。该位清0后停止输出。

			在回环模式下（MCR[4] = 1），输出的break信号由内部回环到接收器，sout输出被强制拉低。 复位值：0x00
5	SP	R/W	Stick Parity 强制产生校验位值 当 PEN、EPS、SP 设置 1，校验位将输出逻辑 0。 当 PEN、SP 设置 1，EPS 置 0，校验位将输出逻辑 1。 写操作，需要 USR[0]为 0，即 UART 外设不在忙状态。 0 = disable 1 = enable 复位值：0x00
4	EPS	R/W	Even Parity Select 选择奇偶校验方式。 写操作，需要 USR[0]为 0，即 UART 外设不在忙状态。 0 = Odd 1 = Even 复位值：0x00
3	PEN	R/W	Parity Enable 奇偶校验使能 写操作，需要 USR[0]为 0，即 UART 外设不在忙状态。 0 = disable 1 = enable 复位值：0x00
2	STOP	R/W	STOP Bits 停止位个数选择。 写操作，需要 USR[0]为 0，即 UART 外设不在忙状态。 0 = 1 stop bit 1 = 当 DLS 中 LCR[1:0]=0 时为 1.5 stop bit，其他值时为 2 stop bit 复位值：0x00
1:0	DLS	R/W	Data Length Select 数据位个数。 写操作，需要 USR[0]为 0，即 UART 外设不在忙状态。 00 = 5 bits 01 = 6 bits 10 = 7 bits 11 = 8 bits 复位值：0x00

14.8.10Modem控制寄存器（MCR）

- **Name:** Modem Control Register
- **Size:** 32 bits
- **Address Offset:** 0x10
- **Read/write access:** read/write

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
预留															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
预留									SIRE	AFCE	LB	OUT2	OUT1	RTS	DTR

Bit	Name	R/W	Description

31:7	预留	-	读操作返回 0
6	SIRE	R/W	SIR Mode Enable 红外模式使能。 写操作，需要 USR[0]为 0，即 UART 外设不在忙状态。 0 = disable 1 = enable 复位值：0x00 如果要是能 SIR 模式,应该在配置 LCR 寄存器前先对 MCR 进行适当的配置。
5	AFCE	R/W	Auto Flow Control Enable 自动流控使能。 0 = disable 1 = enable 复位值：0x00
4	LB	R/W	LookBack Bit 诊断模式使能。该模式用于测试。 在 UART 模式中(MCR[6] = 0),，sout 输出上的数据保持高电平，sout 输出的数据内部回环到 sin 输入。在此模式中所有中断都是可用的。 在回环模式中，modem 控制输入(dsr_n, cts_n, ri_n, dcd_n)不连接，modem 控制输出(dtr_n, rts_n, out1_n, out2_n)均内部回环到 modem 控制输入。 在红外模式中(MCR[6] = 1)，sout 输出保持低电平，sout 输出的电平翻转后内部回环到 sin 输入。 复位值：0x00
3	OUT2	R/W	OUT2 Output2 引脚（out2_n）的输出. 0 = 输出逻辑 1 1 = 输出逻辑 0 在 LookBack 模式中（MCR[4] = 1），out2_n 保持高电平，此时该位置时内部环路的一个输入。 复位值：0x00
2	OUT1	R/W	OUT1 Output1 引脚（out1_n）的输出. 0 = 输出逻辑 1 1 = 输出逻辑 0 在 LookBack 模式中（MCR[4] = 1），out1_n 保持高电平，此时该位置时内部环路的一个输入。 复位值：0x00
1	RTS	R/W	Request to Send RTC 输出。 具体控制方式如下： 1、自动流控未使能（MCR[5]=0）： 该位直接控制 RTS 引脚（Request to Send）的输出。置 1 时 RTS 引脚输出有效电平（低电平），清 0 时 RTS 引脚输出失效电平（高电平）。 2、自动流控使能（MCR[5]=1）且 FIFO 使能（FCR[0] = 1）： 该位作为 RTC 使能位。置 1 时 RTS 引脚输出使能打开，清 0 时 RTS 引脚输出使能关闭。 RTS 引脚输出电平由接收 FIFO 阈值触发控制，当接收数据低于阈值时 RTS 引脚输出有效电平（低电平），当接收数据等于或高于阈值时 RTS 引脚输出无效电平（高电

<table>
<tr><td></td><td></td><td></td><td>平）。
<table>
<tr><td>自动流控</td><td>FIFO</td><td>RTS</td><td>注</td></tr>
<tr><td>使能关闭
MCR[5]=0</td><td>使能关闭
MCR[5]=0</td><td>直接控制</td><td></td></tr>
<tr><td>使能关闭
MCR[5]=0</td><td>使能打开
FCR[0] = 1</td><td>直接控制</td><td></td></tr>
<tr><td>使能打开
MCR[5]=1</td><td>使能关闭
MCR[5]=0</td><td>--</td><td>自动流控模式必须有FIFO 支持</td></tr>
<tr><td>使能打开
MCR[5]=1</td><td>使能打开
FCR[0] = 1</td><td>使能控制</td><td></td></tr>
</table>复位值：0x00</td></tr>
<tr><td>0</td><td>DTR</td><td>R/W</td><td>Data Terminal Ready
数据端就绪态输出
写入寄存器的置与引脚输出的值的逻辑相反
0 = dtr_n 逻辑 1
1 = dtr_n 逻辑 0
数据端就绪的输出用于通知 modem，UART 通信建立完成。
复位值：0x00</td></tr>
</table>

14.8.11Line状态寄存器（LSR）

- **Name:** Line Status Register
- **Size:** 32 bits
- **Address Offset:** 0x14
- **Read/write access:** read-only

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
预留															
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
预留								RFE	TEMT	THRE	BI	FE	PE	OE	DR

Bit	Name	R/W	Description
31:8	预留	-	读操作返回 0
7	RFE	RC_R	Receiver FIFO Error bit 接收 FIFO 错误标志位 在 FIFO 使能的情况下（FCR[0] = 1），该位用于表示接收 FIFO 中至少有一个字节数据存在校验错误，帧错误或 break 状态。 0 = RX FIFO 没有错误 1 = RX FIFO 有错误 在错误字节处于接收 FIFO 顶部且接收 FIFO 中不再有错误字节的数据的情况下，读 LSR 寄存器该位清 0。 复位值：0x00
6	TEMT	R	Transmitter Empty bit ①非 FIFO 模式下（FCR[0] = 0）： 该位为 1 表示发送移位寄存器（TSR）和发送保持寄存器（THR）同时为空。

			②FIFO 模式（FCR[0] = 1）： 该位为 1 表示发送移位寄存器（TSR）和 FIFO 同时为空。 其中： 名词缩写 TSR：Transmitter Shift Register THR：Transmitter Holding Register 不满足以上置位条件，硬件清 0。 复位值：0x01
5	THRE	R	Transmit Holding Register Empty bit. THRE 中断使能，当 THRE 置 1 时会触发 THRE 中断。 ①THRE 中断模式清 0（IER[7] = 0）： 该位用来表示 THR 或发送 FIFO 为空（FIFO 使能有效）。当数据由 THR 或发送 FIFO 输出到 TSR 并且没有新的数据写入到 THR 或发送 FIFO 中时都会导致该位置 1。 ②THRE 中断模式置 1（IER[7] = 1）且 FIFO 使能有效（FCR[0] = 1）： 该位的功能转变为表示发送 FIFO 阈值触发的状态，不再用于表示 THR 为空，发送 FIFO 阈值由 FCR[5:4]设定，当发送 FIFO 中的数据量触发阈值后该位置 1。 不满足以上置位条件，硬件清 0。 复位值：0x01
4	BI	RC_R	Break Interrupt bit Break 中断标志位 该位用来指示 UART 外设接收到对方设备的 break 序列信号数据。如果 UART 外设接收到了 break 信号，break 信号会被认为是每个比特都为 0 的字节接收。若 break 信号保持时间超过 1 次以上传输时间也仅作为 1 个字节数据接收。 在 UART 模式中（MCR[6] = 0），如果输入引脚保持逻辑 0 的时长大于起始位、数据位、校验位和停止位的时间之和，该位置 1。 在 SIR 模式中（MCR[6] = 1），如果输入引脚连续输入的逻辑 0 脉冲时长大于起始位、数据位、校验位和停止位之和，该位置 1。 在非 FIFO 模式中（FCR[0] = 0）： 当 break 信号数据字节被接收后该位马上置 1 在 FIFO 模式中（FCR[0] = 1）： 当 break 信号数据字节被接收且处于队列顶部位置时，该位置 1。 注： 在接收 FIFO 已经满的情况下接收到一个 break 信号，会产生 FIFO 过载。此时，该字节数据的其他附加信息（校验信息，帧错误信息以及 break 信息）都将被丢弃。 读 LSR 寄存器该位清 0。 复位值：0x00
3	FE	RC_R	Framing Error bit 帧错误标志位

			该位用于指示数据接收中发生的帧错误。当 UART 外设接收器没接收到有效的停止位时会产生帧错误。 在 FIFO 模式中，每个字节数据被接收时都会附带一个帧错误信息。产生帧错误的字节数据处于接收 FIFO 顶部时，帧错误标志置 1。 当产生侦错后，UART 外设会尝试重现同步。UART 外设会假设该错误时由于下个字节的起始位引起的并继续接收后续比特位。 需要注意的是，当接收到一个 break 信号数据后帧错误标志会被置 1，也就是说 break 标志置位的同时也会导致帧错误标志置位。因为 break 信号是由连续的逻辑 0 电平表示，所以必定会产生帧错误。 0 = 无帧错误 1 = 有帧错误 读 LSR 寄存器该位清 0。 复位值：0x00
2	PE	RC_R	Parity Error bit 奇偶校验错误标志位 校验使能有效情况下（LCR[3] = 1），该为用于指示数据接收中接收中发生的校验错误。 在 FIFO 模式中，每个字节数据被接收时都会附带一个校验错误信息。产生校验错误的字节数据处于接收 FIFO 顶部时，校验错误标志置 1。 需要注意的是，在校验使能有效（LCR[3] = 1）且校验类型为 Odd（LCR[4] = 0）情况下，当接收到一个 break 信号数据后校验错误标志会被置 1，也就是说 break 标志置位的同时也会导致校验错误标志置位。 0 = 无校验错误 1 = 有校验错误 读 LSR 寄存器该位清 0。 复位值：0x00
1	OE	RC_R	Overrun error bit 过载错误标志位 该位用于指示过载错误。如果新数据被接收而先前数据未及时读取时会产生过载错误。 ①非 FIFO 模式下（FCR[0] = 0）： 1 个新字节数据被接收而在先前在 BRB 中的数据未被及时读取，标志位置 1。如果发生过载，则之前在 BRB 中的数据被覆盖。 ②FIFO 模式下（FCR[0] = 1）： 在接收 FIFO 已满的情况下接收到 1 个新字节数据，会发生过载错误，标志位置 1。如果发生过载，UART 外设会预留先前 FIFO 中接收的数据而丢弃当前接收的数据。 0 = 无过载错误 1 = 有过载错误 读 LSR 寄存器该位清 0。 复位值：0x00
0	DR	R	Data Ready bit 数据有效标志位

			该位用于指示在 BRB 或接收 FIFO 中至少接收到 1 个有效数据。 0 = 没有有效数据 1 = 有有效数据 进行以下操作后，硬件清 0： ①非 FIFO 模式下（FCR[0] = 0），对 BRB 进行读操作。 ②FIFO 模式下（FCR[0] = 1），对接收 FIFO 进行读操作直到接收 FIFO 为空。 复位值：0x00

14.8.12Modem状态寄存器（MSR）

- **Name:** Modem Status Register
- **Size:** 32 bits
- **Address Offset:** 0x18
- **Read/write access:** read-only

0、1、2、3比特用来指示modem控制输入变化，输入发生变化对应位置1。如果IER中modem状态中断使能打开，则会产生相应中断，否则忽略产生的中断。因为在同步modem信号版本的过程中会重置modem控制输入，复位值为0，重置完成后值变为1，所以即使modem中各信号时无效，以上比特位在复位后也会被置1。在复位后对MSR寄存器进行读操作，可以防止不必要的中断产生。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
预留															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
预留								DCD	RI	DSR	CTS	DDCD	TERI	DDSR	DCTS

Bit	Name	R/W	Description
31:8	预留	-	读操作返回 0
7	DCD	R	Data Carrier Detect DCD 状态标志位 该位用于指示当前 modem 控制线 DCD 的状态。当该位置 1,表示 Modem 已经感知到“数据载体”（Data Carrier）。 0 = DCD 输入脚无效（高电平） 1 = DCD 输入脚有效（低电平） 在回环模式中 (MCR[4] = 1), DCD 与 OUT2（MCR[3]）状态相同。 复位值：0x00
6	RI	R	Ring Indicator RI 状态标志位 该位用于指示当前 modem 控制线 RI 的状态。当该位置 1, 表示 Modem 已经接收到“电话响铃信号”（telephone ringing signal）。 0 = RI 输入脚无效（高电平） 1 = RI 输入脚有效（低电平） 在回环模式中 (MCR[4] = 1), RI 与 OUT1（MCR[2]）状态相同。 复位值：0x00

5	DSR	R	Data Set Ready DSR 状态标志位 该位用于指示当前 modem 控制线 DSR 的状态。当该位置 1，表示 Modem 向 UART 外设发送的数据已经准备就绪。 0 = DSR 输入脚无效（高电平） 1 = DSR 输入脚有效（低电平） 在回环模式中 (MCR[4] = 1), DSR 与 DTR（MCR[0]）状态相同。 复位值：0x00
4	CTS	R	Clear to Send CTS 状态标志位 该位用于指示当前 modem 控制线 CTS 的状态。当该位置 1，表示 UART 外设接收到 Modem 的请求数据信号(RTS)。 0 = CTS 输入脚无效（高电平） 1 = CTS 输入脚有效（低电平） 在回环模式中 (MCR[4] = 1), CTS 与 RTS（MCR[1]）状态相同。 复位值：0x00
3	DDCD	RC_R	Delta Data Carrier Detect DDCD 状态标志位 该位用于指示当前 modem 控制线 DCD 的状态发生变化。该位置 1，表示上次 MSR 读操作后，DCD 状态发生变化。 0 = 上次 MSR 读操作后，DCD 的状态未发生变化 1 = 上次 MSR 读操作后，DCD 的状态发生变化 对 MSR 读操作清 0 该位，在回环模式中 (MCR[4] = 1)，DDCD 指示 OUT2（MCR[3]）的状态变化。 注，以下情况 DDCD 会置 1： DDCD 状态为 0，DCD 信号有效且复位产生，如果 DCD 一直保持有效，直到复位完成，则 DDCD 会置 1。 复位值：0x00
2	TERI	RC_R	Trailing Edge of Ring Indicator TERI 状态标志位 该位用于指示当前 modem 控制线 RI 的状态发生变化（由低电平有效状态变为高电平无效状态）。该位置 1，表示上次 MSR 读操作后，RI 状态发生变化。 0 = 上次 MSR 读操作后，RI 的状态未发生变化 1 = 上次 MSR 读操作后，RI 的状态发生变化 对 MSR 读操作清 0 该位，在回环模式中 (MCR[4] = 1)，TERI 指示 OUT1（MCR[2]）的状态变化（由低电平有效状态变为高电平无效状态）。 复位值：0x00
1	DDSR	RC_R	Delta Data Set Ready DDSR 状态标志位 该位用于指示当前 modem 控制线 DSR 的状态发生变化。该位置 1，表示上次 MSR 读操作后，DSR 的状态发生变化。 0 = 上次 MSR 读操作后，DSR 的状态未发生变化 1 = 上次 MSR 读操作后，DSR 的状态发生变化 对 MSR 读操作清 0 该位，在回环模式中 (MCR[4] = 1)，

			DDSR 指示 DTR（MCR[0]）的状态变化。 注，以下情况 DDSR 会置 1： DDSR 状态为 0，DSR 信号有效且复位产生，如果 DSR 一直保持有效，直到复位完成，则 DDSR 会置 1。 复位值：0x00
0	DCTS	RC_R	Delta Clear to Send DCTS 状态标志位 该位用于指示当前 modem 控制线 CTS 的状态发生变化。该位置 1，表示上次 MSR 读操作后，CTS 的状态发生变化。 0 = 上次 MSR 读操作后，CTS 的状态未发生变化 1 = 上次 MSR 读操作后，CTS 的状态发生变化 对 MSR 读操作清 0 该位，在回环模式中 (MCR[4] = 1)，DCTS 指示 CTS（MCR[1]）的状态变化。 注，以下情况 DCTS 会置 1： DCTS 状态为 0，CTS 信号有效且复位产生，如果 CTS 一直保持有效，直到复位完成，则 DCTS 会置 1。 复位值：0x00

14.8.13FIFO访问使能寄存器（FAR）

- **Name:** FIFO Access Register
- **Size:** 32 bits
- **Address Offset:** 0x70
- **Read/write access:** read/write

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
预留															
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
预留															FAR

Bit	Name	R/W	Description
31:1	预留	-	-
0	FAR	R/W	FIFO Access Register FIFO 访问使能 该位用于用于 FIFO 测试，控制 FIFO 是否可以被用户访问。当使能打开后，用户可以写接收 FIFO，读发送 FIFO。使能关闭，用户只能通过 RBR 和 THR 来访问 FIFO。】 0 = FIFO 访问使能关闭 1 = FIFO 访问使能打开 注：当 FIFO 访问使能进行打开/关闭操作时，接收和发送 FIFO 的控制部分会复位并且 FIFO 也被视为空。 复位值：0x00

14.8.14读发送FIFO寄存器（TFR）

- **Name:** Transmit FIFO Read
- **Size:** 32 bits
- **Address Offset:** 0x74
- **Read/write access:** read-only

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
预留															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
预留								TFRD							

Bit	Name	R/W	Description
31:8	预留	-	-
7:0	TFRD	R	Transmit FIFO Read 读发送 FIFO 数据 该位只有当 FIFO 访问使能打开后操作才有效（FAR[0] = 1）。 当 FIFO 功能打开，对该位进行读操作将返回发送 FIFO 队列顶部数据。进行连续的读操作将依次取出 FIFO 中的数据，每次读操作读取的数据均是当前发送 FIFO 队列顶部数据。 当 FIFO 功能关闭，对该位的读操作将返回 THR 内的数据。 复位值：0x00

14.8.15写接收FIFO寄存器（RFW）

- **Name:** Receive FIFO Write
- **Size:** 32 bits
- **Address Offset:** 0x78
- **Read/write access:** write-only

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
预留															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
预留						RFFE	RFPE	RFWD							

Bit	Name	R/W	Description
31:10	预留	-	预留
9	RFFE	W	Receive FIFO Framing Error. 接收 FIFO 帧错误命令 该位只有当 FIFO 访问使能打开后操作才有效（FAR[0] = 1）。 当 FIFO 功能打开，该位用于写帧错误信息到接收 FIFO。 当 FIFO 功能关闭，该位用于写帧错误信息到 RBR。 复位值：0x00
8	RFPE	W	Receive FIFO Parity Error. 接收 FIFO 校验错误命令 该位只有当 FIFO 访问使能打开后操作才有效（FAR[0] = 1）。 当 FIFO 功能打开，该位用于写校验错误信息到接收 FIFO。 当 FIFO 功能关闭，该位用于写校验错误信息到 RBR。 复位值：0x00
7:0	RFWD	W	Receive FIFO Write Data 写接收 FIFO 数据

			该位只有当 FIFO 访问使能打开后操作才有效（FAR[0] = 1）。 当 FIFO 功能打开，写入该位的数据将压到接收 FIFO 中。每次写入该位的数据将在下次写该位时被压到接收 FIFO 中。 当 FIFO 功能关闭，写入该位的数据将压到 RBR 中。 复位值：0x00

14.8.16UART状态寄存器（USR）

- **Name:** UART Status Register
- **Size:** 32 bits
- **Address Offset:** 0x7C
- **Read/write access:** read-only

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
预留															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
预留											RFF	RFNE	TFE	TFNF	BUSY

Bit	Name	R/W	Description
31:5	预留	-	预留
4	RFF	R	Receive FIFO Full 接收 FIFO 满（completely full）标志。 0 = 接收 FIFO 未满 1 = 接收 FIFO 满 当接收 FIFO 不再是满的时候该位被清空。 复位值：0x00
3	RFNE	R	Receive FIFO Not Empty 接收 FIFO 非空标志。 0 = 接收 FIFO 空 1 = 接收 FIFO 非空 当接收 FIFO 为空时该位被清空。 复位值：0x00
2	TFE	R	Transmit FIFO Empty 发送 FIFO 为空（completely empty）标志。 0 = 发送 FIFO 非空 1 = 发送 FIFO 空 当发送 FIFO 为非空时该位被清空。 复位值：0x00
1	TFNF	R	Transmit FIFO Not Full 发送 FIFO 未满标志。 0 = 发送 FIFO 已满 1 = 发送 FIFO 未满 当发送 FIFO 为已满时该位被清空。 复位值：0x00
0	BUSY	R	UART Busy UART 忙标志位 该位为 1 串行传输正在进行，为 0 表示 UART 外设空闲或不活动。

			0 = UART 外设空闲或不活动 1 = UART 外设忙（正在进行串行数据传输） 以下任意一种情况将导致该位置 1，即 UART 外设忙： 1、串口接口正在进行发送 2、FIFO 访问使能未打开（FAR[0] = 0），波特率分频不为 0（DLH,DLL ≠ 0），分频访问使能关闭（LCR.DLAB = 0）且 THR 中有发送数据 3、串口正在进行数据接收 4、FIFO 访问使能未打开（FAR[0] = 0）且 RBR 中有接收的数据 复位值：0x00

14.8.17发送FIFO数据量（TFL）

- **Name:** Transmit FIFO Level
- **Size:32bits**
- **Address Offset:** 0x80
- **Read/write access:** read-only

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
预留															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
预留												Transmit FIFO Level			

Bit	Name	R/W	Description
31:4	预留	-	预留
3:0	TFL	R	Transmit FIFO Level 当前发送 FIFO 中数据的个数。 复位值：0x00

14.8.18接收FIFO数据量（RFL）

- **Name:** Receive FIFO Level
- **Size:32bits**
- **Address Offset:** 0x84
- **Read/write access:** read-only

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
预留															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
预留												Receive FIFO Level			

Bit	Name	R/W	Description
31:4	预留	-	预留
3:0	RFL	R	Receive FIFO Level 当前接收 FIFO 中数据的个数。 复位值：0x00

14.8.19软复位寄存器（SRR）

- **Name:** Software Reset Register
- **Size:** 32 bits
- **Address Offset:** 0x88
- **Read/write access:** write-only

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
预留															
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
预留													XFR	RFR	UR

Bit	Name	R/W	Description
31:3	预留	-	预留
2	XFR	W	XMIT FIFO Reset. 发送 FIFO 复位 该操作位是 FCR[2]的影子操作位。当用户只复位发送 FIFO 时，通过 FCR 寄存器复位发送 FIFO 会覆盖之前的 FCR 寄存器设置，使用该影子操作位可以避免上述情况。该操作会复位发送 FIFO 的控制部分并初始发送 FIFO 为空。复位会使 DMA TX 请求信号无效。该位硬件清 0。 复位值：0x00
1	RFR	W	RCVR FIFO Reset 接收 FIFO 复位 该操作位是 FCR[1]的影子操作位。当用户只复位接收 FIFO 时，通过 FCR 寄存器复位接收 FIFO 会覆盖之前的 FCR 寄存器设置，使用该影子操作位可以避免上述情况。该操作会复位接收 FIFO 的控制部分并初始接收 FIFO 为空。复位会使 DMA TX 请求信号无效。该位硬件清 0。 复位值：0x00
0	UR	W	UART Reset UART 复位 该位操作复位 UART 外设，需要经过 2 个执行时钟周期，等待 pclk 和 sclk 都完成复位。复位完成后立即清除复位标志。 复位值：0x00

14.8.20发送请求影子寄存器（SRTS）

- **Name:** Shadow Request to Send
- **Size:** 32 bits
- **Address Offset:** 0x8C
- **Read/write access:** read/write

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
预留															
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
预留															SRTS

<table>
<tr><th>Bit</th><th>Name</th><th>R/W</th><th>Description</th></tr>
<tr><td>31:1</td><td>预留</td><td>-</td><td>预留</td></tr>
<tr><td>0</td><td>SRTS</td><td>R/W</td><td>Shadow Request to Send
RTS 输出影子寄存器

该操作位是 MCR[1](RTS)的影子操作位。
该位用于控制 UART 外设 RTC 输出，具体控制方式如下：
1、自动流控未使能（MCR[5]=0）：
该位直接控制 RTS 引脚（Request to Send）的输出。置 1 时 RTS 引脚输出有效电平（低电平），清 0 时 RTS 引脚输出失效电平（高电平）。
2、自动流控使能（MCR[5]=1）且 FIFO 使能（FCR[0] = 1）：
该位作为 RTC 使能位。置 1 时 RTS 引脚输出使能打开，清 0 时 RTS 引脚输出使能关闭。
RTS 引脚输出电平由接收 FIFO 阈值触发控制，当接收数据低于阈值时 RTS 引脚输出有效电平（低电平），当接收数据等于或高于阈值时 RTS 引脚输出无效电平（高电平）。

<table>
<tr><th>自动流控</th><th>FIFO</th><th>RTS</th><th>注</th></tr>
<tr><td>使能关闭
MCR[5]=0</td><td>使能关闭
MCR[5]=0</td><td>直接控制</td><td></td></tr>
<tr><td>使能关闭
MCR[5]=0</td><td>使能打开
FCR[0] = 1</td><td>直接控制</td><td></td></tr>
<tr><td>使能打开
MCR[5]=1</td><td>使能关闭
MCR[5]=0</td><td>--</td><td>自动流控模式必须有 FIFO 支持</td></tr>
<tr><td>使能打开
MCR[5]=1</td><td>使能打开
FCR[0] = 1</td><td>使能控制</td><td></td></tr>
</table>
复位值：0x00</td></tr>
</table>

14.8.21Break信号控制影子寄存器（SBCR）

- **Name:** Shadow Break Control Register
- **Size:** 32 bits
- **Address Offset:** 0x90
- **Read/write access:** read/write

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
预留															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
预留															SBCR

Bit	Name	R/W	Description
31:1	预留	-	预留
0	SBCR	R/W	Shadow Break Control Bit Break控制影子寄存器 该操作位是 LCR[6](Break Control bit)的影子操作位。通过该位可以省去对 LCR 的“读-修改-写”操作。

			该位用于产生输出break信号到接收设备，该位置1，UART外设输出引脚会强行输出逻辑0信号。 当在非回环模式中（MCR[4] = 0），UART模式（MCR[4] = 0），sout输出将强制拉低直。红外模式（MCR[4] = 1），sout持续输出脉冲信号。该位清0后停止输出。 当在回环模式下（MCR[4] = 1），输出的break信号由内部回环到接收器，sout输出被强制拉低。 复位值：0x00

14.8.22DMA模式影子寄存器（SDMAM）

- **Name:** Shadow DMA Mode
- **Size:** 32 bits
- **Address Offset:** 0x94
- **Read/write access:** read/write

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
预留															
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
预留															SDMAM

Bit	Name	R/W	Description
31:1	预留	-	预留
0	SDMAM	R/W	Shadow DMA Mode. DMA 模式影子寄存器 该操作位是 FCR[3](DMA mode bit)的影子操作位。 该位可以避免在只修改 DMA 模式位的情况下，影响 FCR 之前设置的内容。 该位在额外DMA握手信号没有选择（DMA_EXTRE = NO）的情况下用于确定DMA发送请求和接收请求信号的模式。 0 = 不使用DMA 1 = 使用DMA 复位值：0x00

14.8.23FIFO使能影子寄存器（SFE）

- **Name:** Shadow FIFO Enable
- **Size:** 32 bits
- **Address Offset:** 0x98
- **Read/write access:** read/write

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
预留															
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
预留															SFE

Bit	Name	R/W	Description
31:1	预留	-	预留
0	SFE	R/W	Shadow FIFO Enable

			FIFO 使能影子寄存器 该操作位是 FCR[0](FIFO enable bit)的影子操作位。 该位可以避免在只修改 FIFO 使能位的情况下，影响 FCR 之前设置的内容。 用于使能接收和发送FIFO。当该位发送变化时接收和发送FIFO都将复位。 复位值：0x00

14.8.24接收FIFO满阈值设置影子寄存器（SRT）

- **Name:** Shadow RCVR Trigger
- **Size:** 32 bits
- **Address Offset:** 0x9C
- **Read/write access:** read/write

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
预留															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
预留														SRT	

Bit	Name	R/W	Description
31:2	预留	-	预留
1:0	SRT	R/W	Shadow RCVR Trigger 接收FIFO满触发的阈值设定影子寄存器。 该操作位是 FCR[7:6](RCVR Trigger)的影子操作位。 该位可以避免在只修改接收 FIF 阈值的情况下，影响 FCR 之前设置的内容。 当接收FIFO中的数据超过设定的阈值后，会触发接收数据有效中断。 在自动流控模式下，该阈值用于决定接收rts_n信号状态（在RTC_FCT使能关闭的状态下）。 在DMA模式下，FIFO中数据量触发阈值后UART发出DMA请求。 以下为支持的阈值类型。 00 – 1 character in the FIFO 01 – FIFO ¼ full 10 – FIFO ½ full 11 – FIFO 2 less than full 复位值：0x00

14.8.25发送FIFO空阈值设置影子寄存器（STET）

- **Name:** Shadow TX Empty Trigger
- **Size:** 32 bits
- **Address Offset:** 0xA0
- **Read/write access:** read/write

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
预留															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

预留	STET

Bit	Name	R/W	Description
31:2	预留	-	预留
1:0	STET	R/W	Shadow TX Empty Trigger 发送FIFO空的阈值设定影子寄存器。 该操作位是 FCR[5:4](TX Empty Trigger)的影子操作位。该位可以避免在只修改发送 FIF 阈值的情况下，影响 FCR 之前设置的内容。 当PTIME使能打开(IER[7] = 1)，发送FIFO中数据等于或低于该阈值将触发THRE中断产生。 在DMA模式下，FIFO中数据量触发阈值后UART发出DMA请求。 以下为支持的阈值类型。 00 – FIFO empty 01 – 2 characters in the FIFO 10 – FIFO ¼ full 11 – FIFO ½ full 复位值：0x00

14.8.26发送暂停寄存器（HTX）

- **Name:** Halt TX
- **Size:** 32 bits
- **Address Offset:** 0xA4
- **Read/write access:** read/write

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
预留															
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
预留															HTX

Bit	Name	R/W	Description
31:1	预留	-	-
0	HTX	R/W	Halt TX 发送暂停寄存器 该寄存器用于 UART 测试，对其操作来停止 UART 发送。这样在 FIFO 使能打开的状态下，发送 FIFO 可以被用户写满。 0 = 停止 TX 使能关闭 1 = 停止 TX 使能打开 复位值：0x00

14.8.27DMA软应答（DMASA）

- **Name:** DMA Software Acknowledge
- **Size:** 32 bits
- **Address Offset:** 0xA8
- **Read/write access:** write

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
预留															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
预留															DMASA

Bit	Name	R/W	Description
31:1	预留	-	预留
0	DMASA	W	DMA Software Acknowledge DMA软应答 如果在DMA传输过程中产生了错误需要终止，那么可以是用该寄存器产生一个DMA软应答。 例如，如果DMA通道使能关闭，UART需要清除它的请求信号，那么对该寄存器的操作将使TX request, TX single, RX request 和 RX single 信号无效。对该寄存器的操作无需用户清0，硬件自行清0。 复位值：0x00

15 I2C接口

15.1 I2C简介

I2C(芯片间)总线接口连接微控制器和串行I2C总线。它提供多主机功能，控制所有I2C总线特定的时序、协议、仲裁和定时。支持标准和快速两种模式。

15.2 I2C主要特点

- I2C时钟由PCLK提供，即I2C_CLK = PCLK
- 多主机功能：该模块既可做主设备也可做从设备
- 包含硬件实现收发FIFO，深度为8
- I2C从/主设备功能支持
- 产生和检测7位/10位地址和广播呼叫
- 支持不同的通讯速度
  - 标准速度(高达100 kHz)
  - 快速(高达400 kHz)
- 支持I2C协议SDA HOLD/SETUP时间设定

15.3 I2C功能描述

I2C模块接收和发送数据，并将数据从串行转换成并行，或并行转换成串行。可以开启或禁止中断。接口通过数据引脚(SDA)和时钟引脚(SCL)连接到I2C总线。允许连接到标准(高达100kHz)或快速(高达400kHz)的I2C总线。

15.3.1 I2C外设时钟（I2C_CLK）

I2C外设时钟由PCLK提供，不分频，即I2C_CLK = PCLK

定义如下参数：
I2C_CLK：I2C外设时钟频率
I2C_CLK_PERIOD：I2C外设时钟周期时间

15.3.2 接收/发送FIFO

I2C外设包含2个独立的深度为8的接收和发送FIFO。CPU对寄存器IC_DATA_CMD写操作，数据写入发送FIFO，CPU对寄存器IC_DATA_CMD读操作，数据由接收FIFO中取出。

15.3.3 START和STOP信号产生

I2C外设工作在主模式下，当CPU向IC_DATA_CMD数据寄存器中写数据时，I2C产生START信号。当CPU向IC_DATA_CMD中写入数据的IC_DATA_CMD[9]位为1会使I2C产生STOP信号。在发送的过程中IC_DATA_CMD[9]不置“1”，I2C不会产生STOP信号，即使发送FIFO已空。

I2C外设工作在从模式下，不会产生START和STOP信号。

15.3.4 I2C协议

主模式时，I2C接口启动数据传输并产生时钟信号。串行数据传输总是以起始条件开始并以停止条件结束。起始条件和停止条件都是在主模式下由软件控制产生。

从模式时，I2C接口能识别它自己的地址(7位或10位)和广播呼叫地址。软件能够控制开启或禁止广播呼叫地址的识别。

数据和地址按8位/字节进行传输，高位在前。跟在起始条件后的1或2个字节是地址(7位模式为1个字节，10位模式为2个字节)。地址只在主模式发送。

在一个字节传输的8个时钟后的第9个时钟期间，接收器必须回送一个应答位(ACK)给发送器。参考下图。

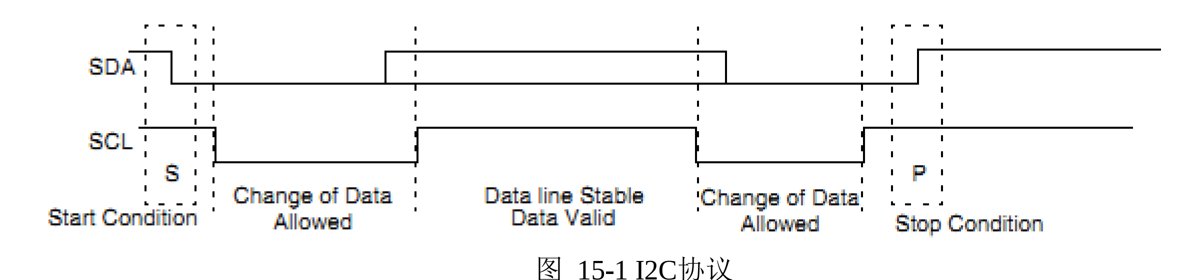


图 15-1 I2C协议

15.3.5 I2C主模式

操作流程如下：

1、 IC_ENABLE[0] = 0关闭I2C外设使能
2、 通过 IC_CON 设定工作速度模式，设置工作模式为主模式
3、 写 IC_TAR 设定目标从设备地址，可根据 IC_TAR[12]设定地址模式为 7 地址或 10 地址
4、 根据工作速度模式设定 IC_SS_SCL_HCNT/IC_SS_SCL_LCNT 或 IC_FS_SCL_HCNT/IC_FS_SCL_LCNT 波特率
5、 根据需要设定 IC_SDA_SETUP/IC_SDA_HOLD 参数
6、 IC_ENABLE[0] = 1 打开 I2C 外设使能

15.3.6 I2C从模式

1、 IC_ENABLE[0] = 0 关闭 I2C 外设使能
2、 通过 IC_CON 设置工作模式为从模式，设定从地址响应模式（7 地址或 10 地址）
3、 写 IC_SAR 设定本机从设备地址
4、 IC_ENABLE[0] = 1 打开 I2C 外设使能

15.3.7 I2C毛刺抑制

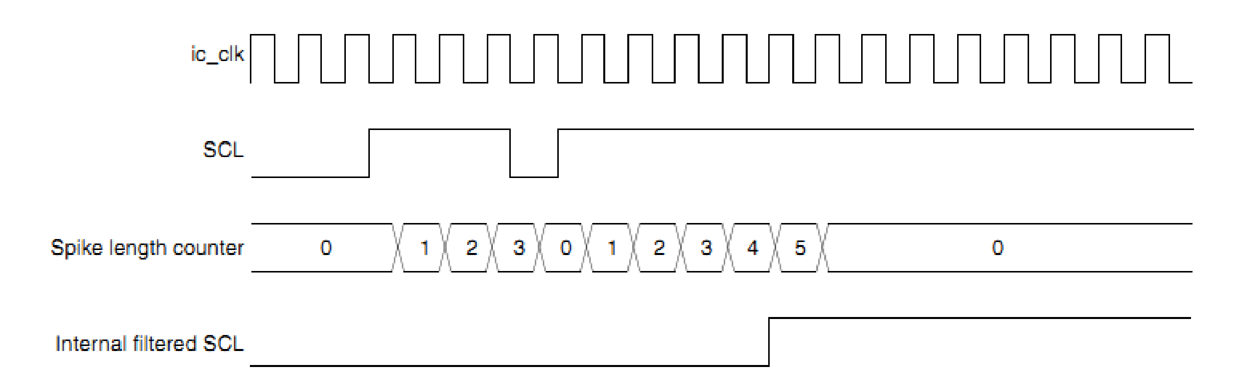
I2C外设通过IC_FS_SPKLEN寄存器抑制I2C总线上毛刺。

I2C外设包含可编程的毛刺抑制单元，该单元用于配合“标准”、“快速”I2C模式的要求。毛刺抑制器使用计数器监控SCL和SDA电平状态，在输入电平被采样前会检查电平是否保持了预设数量的IC_CLK周期。SCL和SDA信号有各自独立的计数器。用户可以编程设定预设数量，在计算预设数量时需要考虑IC_CLK频率及尖峰宽度。

当电平发生变化时就会使各自的计数器启动。具体情况如下：

1. 计数器累计达到设定值之前，输入电平信号一直保持稳定不变。此时内部电平信号状态更新为新输入状态，同时毛刺抑制计数器复位并停止。计数器直到下个电平发送变化后再次启动。
2. 计数器累计达到设定值之前，输入电平信号发生变化。此时内部电平信号状态不发生变化，同时毛刺抑制计数器复位并停止。计数器直到下个电平发送变化后再次启动。

抑制过程如下图：



毛刺抑制器寄存器宽度为8。该寄存器只能在I2C外设被禁用时才能操作。最小写入值为1，若尝试写入小于1的值，结果默认设定为1。

抑制毛刺宽度 = IC_FS_SPKLEN* I2C_CLK_PERIOD

15.3.8 I2C波特率设定

I2C外设每种速度模式包含2个波特率设定寄存器
标准模式波特率设定寄存器：
IC_SS_SCL_HCNT
IC_SS_SCL_LCNT
快速模式波特率设定寄存器：
IC_FS_SCL_HCNT
IC_FS_SCL_LCNT

IC_xx_SCL_HCNT：波特率高电平周期计数
IC_xx_SCL_LCNT：波特率低电平周期计数

定义以下参数：
SCL_PERIOD：波特率周期时间
SCL_PERIOD_Ht：波特率高电平保持时间
SCL_PERIOD_Lt：波特率低电平保持时间

根据以上定义则计算如下：
SCL_PERIOD = SCL_PERIOD_Ht + SCL_PERIOD_Lt
SCL_PERIOD_Ht = IC_xx_SCL_HCNT * I2C_CLK_PERIOD
SCL_PERIOD_Lt = IC_xx_SCL_LCNT * I2C_CLK_PERIOD

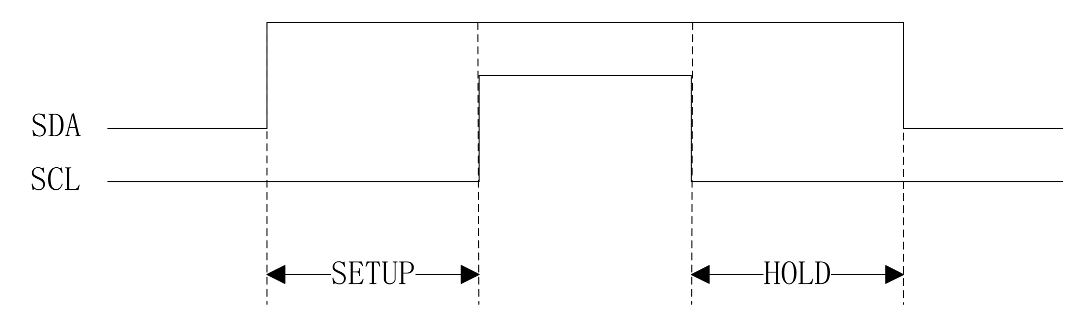
SCL_PERIOD = （IC_xx_SCL_HCNT + IC_xx_SCL_LCNT）* I2C_CLK_PERIOD

波特率高/低电平最小值要求：
IC_SS_SCL_LCNT/ IC_FS_SCL_LCNT必须大于IC_FS_SPKLEN +7
IC_SS_SCL_HCNT / IC_FS_SCL_HCNT必须大于IC_FS_SPKLEN +5

15.3.9 SDA SETUP/HOLD时间设定

I2C外设提供IC_SDA_SETUP和IC_SDA_HOLD寄存器对SDA SETUP和HOLD时间进行设定

SETUP/HOLD示意图：



关于SDA SETUP和HOLD详细时间参数要求见“I2C总线协议规范”。

IC_SDA_SETUP：SETUP时间周期计数
IC_SDA_HOLD：HOLD时间周期计数

SETUP时间 = IC_SDA_SETUP * I2C_CLK_PERIOD
HOLD时间 = IC_SDA_HOLD* I2C_CLK_PERIOD

15.4 I2C寄存器描述

15.4.1 地址映射表

I2C基地址列表

地址范围	基地址	外设	总线
0x4001_1000-0x4001_1FFF	0x4001_1000	I2C0	APB0

表 15-1 I2C寄存器表

偏移地址	寄存器名称	宽度（bit）	复位值
0x00	IC_CON	32	0x0000007D
0x04	IC_TAR	32	0x00001055
0x08	IC_SAR	32	0x00000055
0x0C	IC_HS_MADDR	32	0x00000000
0x10	IC_DATA_CMD	32	0x00000000
0x14	IC_SS_SCL_HCNT	32	0x00000190
0x18	IC_SS_SCL_LCNT	32	0x000001D6
0x1C	IC_FS_SCL_HCNT	32	0x0000003C
0x20	IC_FS_SCL_LCNT	32	0x00000082
0x24	保留	32	0x00000000
0x28	保留	32	0x00000000
0x2C	IC_INTR_STAT	32	0x00000000
0x30	IC_INTR_MASK	32	0x000008FF
0x34	IC_RAW_INTR_STAT	32	0x00000000
0x38	IC_RX_TL	32	0x00000000
0x3C	IC_TX_TL	32	0x00000000
0x40	IC_CLR_INTR	32	0x00000000
0x44	IC_CLR_RX_UNDER	32	0x00000000
0x48	IC_CLR_RX_OVER	32	0x00000000
0x4C	IC_CLR_TX_OVER	32	0x00000000
0x50	IC_CLR_RD_REQ	32	0x00000000
0x54	IC_CLR_TX_ABRT	32	0x00000000
0x58	IC_CLR_RX_DONE	32	0x00000000
0x5C	IC_CLR_ACTIVITY	32	0x00000000

0x60	IC_CLR_STOP_DET	32	0x00000000
0x64	IC_CLR_START_DET	32	0x00000000
0x68	IC_CLR_GEN_CALL	32	0x00000000
0x6C	IC_ENABLE	32	0x00000000
0x70	IC_STATUS	32	0x00000006
0x74	IC_TXFLR	32	0x00000000
0x78	IC_RXFLR	32	0x00000000
0x7C	IC_SDA_HOLD	32	0x00000001
0x80	IC_TX_ABRT_SOURCE	32	0x00000000
0x84	IC_SLV_DATA_NACK_ONLY	32	0x00000000
0x88	IC_DMA_CR	32	0x00000000
0x8C	IC_DMA_TDLR	32	0x00000000
0x90	IC_DMA_RDLR	32	0x00000000
0x94	IC_SDA_SETUP	32	0x00000064
0x98	IC_ACK_GENERAL_CALL	32	0x00000001
0x9C	IC_ENABLE_STATUS	32	0x00000000
0xA0	IC_FS_SPKLEN	32	0x00000005

15.4.2 I2C控制寄存器（IC_CON）

- **Name: I2C Control Register**
- **Size: 32bits**
- **Address Offset: 0x00**

该寄存器写操作必须在I2C外设关闭的状态下进行，其他情况写无效。

<table>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td></tr>
<tr><td colspan="8">保留</td></tr>
<tr><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td colspan="8">保留</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td></tr>
<tr><td colspan="8">保留</td></tr>
<tr><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td>保留</td><td>IC_SLAVE_DISABLE</td><td>IC_RESTART_EN</td><td>IC_10BITADDR_MASTER</td><td>IC_10BITADDR_SLAVE</td><td colspan="2">SPEED</td><td>MASTER_MODE</td></tr>
</table>

Bit	Name	R/W	Description
31:7	保留	-	
6	IC_SLAVE_DISABLE	R/W	I2C 从模式使能位。 0-从模式打开 1-从模式关闭
5	IC_RESTART_EN	R/W	I2C 主模式中 RESTART 信号支持 0-不支持 RESTART 信号 1-支持 RESTART 信号
4	IC_10BITADDR_MASTER	R	I2C 主模式中地址为模式 I2C 主模式中地址为模式由 IC_TAR[12]设定，该位仅用于获取 IC_TAR[12]状态 0-7bit 地址模式

			1-10bit 地址模式
3	IC_10BITADDR_SLAVE	R/W	I2C 从模式中地址位模式 0-7bit 地址模式 1-10bit 地址模式
2:1	SPEED	R/W	I2C 传输速率模式 1-标准模式（standard mode）(0 到 100kbit/s) 2-快速模式（fast mode）(400kbit/s)
0	MASTER_MODE	R/W	I2C 主模式使能位 0-主模式关闭 1-主模式打开

I2C主从模式配置表：

IC_SLAVE_DISABLE (IC_CON[6])	MASTER_MODE IC_CON[0]	State
0	0	Slave Device
0	1	Config Error
1	0	Config Error
1	1	Master Device

15.4.3 I2C目标地址寄存器（IC_TAR）

- **Name: I2C Target Address Register**
- **Size: 32 bits**
- **Address Offset: 0x04**
- **Read/Write Access: Read/Write**

该寄存器仅在I2C外设关闭的状态（IC_ENABLE[0]=0）或I2C空闲状态（IC_ENABLE[0]=1 and IC_STATUS[5]=0 and IC_CON[0]=1 and IC_STATUS[2]=1）写操作，其他情况写无效。

31	30	29	28	27	26	25	24
保留							
23	22	21	20	19	18	17	16
保留							
15	14	13	12	11	10	9	8
保留			IC_10BITADDR_MASTER	SPECIAL	GC_OR_START	IC_TAR	
7	6	5	4	3	2	1	0
IC_TAR							

Bit	Name	R/W	Description
31:13	保留	-	
12	IC_10BITADDR_MASTER	R/W	I2C 主模式中地址为模式 0-7bit 地址模式 1-10bit 地址模式
11	SPECIAL	R/W	I2C 地址呼叫(General Call)和起始字节(START BYTE)命令支持 0-不支持 1-支持
10	GC_OR_START	R/W	I2C 地址呼叫(General Call)/起始字节(START BYTE)选择位 0-地址呼叫 1=起始字节

			地址呼叫：发出地址呼叫后，通过读取 IC_RAW_INTR_STAT 寄存器中 TX_ABRT 位获取呼叫结果。地址呼叫模式会一直保持到 SPECIAL 清“0”为止。
9:0	IC_TAR	R/W	I2C 目标地址 I2C 工作在主模式，从设备目标地址。 I2C 发出地址呼叫时忽略 IC_TAR，发出起始字节时，CPU 需要对 IC_TAR 进行 1 次写操作。

15.4.4 I2C从地址寄存器（IC_SAR）

- **Name: I2C Slave Address Register**
- **Size: 32 bits**
- **Address Offset: 0x08**
- **Read/Write Access: Read/Write**

该寄存器写操作必须在I2C外设关闭的状态下进行，其他情况写无效。

31	30	29	28	27	26	25	24
保留							
23	22	21	20	19	18	17	16
保留							
15	14	13	12	11	10	9	8
						IC_SAR	
7	6	5	4	3	2	1	0
IC_SAR							

Bit	Name	W/R	Description
31:10	保留	-	
9:8	IC_SAR	W/R	I2C 从地址 I2C 工作在从模式状态，I2C 总线上主设备操作时匹配用地址。

15.4.5 I2C接收/发送数据命令寄存器（IC_DATA_CMD）

- **Name: Rx/Tx Data Buffer and Command Register**
- **Size: 32 bits**
- **Address Offset: 0x10**
- **Read/Write Access: Read/Write**

该寄存器写操作必须在I2C外设关闭的状态下进行，其他情况写无效。

31	30	29	28	27	26	25	24
保留							
23	22	21	20	19	18	17	16
保留							
15	14	13	12	11	10	9	8
保留					RESTART	STOP	CMD
7	6	5	4	3	2	1	0
DATA							

Bit	Name	W/R	Description

31:11	保留	-	
10	RESTART	W	RESTART 信号控制位 在当前字节发送之后或下个字节接收之前指定是否产生 RESTART 信号。 当 IC_CON 中 IC_RESTART_EN 为"1"时对该位操作有效。 1-产生 RESTART 信号 0-仅当传输方向发生变化时产生 RESTART 信号 IC_CON 中 IC_RESTART_EN = 1,I2C 产生 RESTART。 IC_RESTART_EN = 0,RESTART 信号由 STOP + START 信号代替。
9	STOP	W	STOP 信号控制位 在当前字节发送之后或下个字节接收之前指定是否产生 STOP 信号。 1-产生 STOP 信号 0-不产生 STOP 信号
8	CMD	W	I2C 读写控制位 1=读操作 0=写操作
9:8	DATA	W/R	I2C 数据域 写操作数据将被放入 TX FIFO 中，读操作由 RX FIFO 取数据。

15.4.6 I2C标准速率模式SCL高电平计数器（IC_SS_SCL_HCNT）

- **Name: Standard Speed I2C Clock SCL High Count Register**
- **Size: 32 bits**
- **Address Offset: 0x14**
- **Read/Write Access: Read/Write**

该寄存器写操作必须在I2C外设关闭的状态下进行，其他情况写无效。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IC_SS_SCL_HCNT															

Bit	Name	W/R	Description
31:16	保留	-	
15:0	IC_SS_SCL_HCNT	W/R	I2C SCL 高电平周期计数器 I2C 工作在标准速率（standard speed）模式下，该寄存器值决定 SCL 高电平保持时间，寄存器最小值为 6。

15.4.7 I2C标准速率模式SCL低电平计数器（IC_SS_SCL_LCNT）

- **Name: Standard Speed I2C Clock SCL Low Count Register**
- **Size: 32 bits**
- **Address Offset: 0x18**
- **Read/Write Access: Read/Write**

该寄存器写操作必须在I2C外设关闭的状态下进行，其他情况写无效。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IC_SS_SCL_LCNT															

Bit	Name	W/R	Description
31:16	保留	-	
15:0	IC_SS_SCL_LCNT	W/R	I2C SCL 低电平周期计数器 I2C 工作在标准速率（standard speed）模式下，该寄存器值决定 SCL 低电平保持时间，寄存器最小值为 6。

15.4.8 I2C快速模式SCL高电平计数器（IC_FS_SCL_HCNT）

- **Name: Fast Speed I2C Clock SCL High Count Register**
- **Size: 32 bits**
- **Address Offset: 0x1c**
- **Read/Write Access: Read/Write**

该寄存器写操作必须在I2C外设关闭的状态下进行，其他情况写无效。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IC_FS_SCL_HCNT															

Bit	Name	W/R	Description
31:16	保留	-	
15:0	IC_FS_SCL_HCNT	W/R	I2C SCL 高电平周期计数器 I2C 工作在高速（fast speed）模式下，该寄存器值决定 SCL 高电平保持时间，寄存器最小值为 8。

15.4.9 I2C快速模式SCL低电平计数器（IC_FS_SCL_LCNT）

- **Name: Fast Speed I2C Clock SCL Low Count Register**
- **Size: 32bits**
- **Address Offset: 0x20**
- **Read/Write Access: Read/Write**

该寄存器写操作必须在I2C外设关闭的状态下进行，其他情况写无效。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IC_FS_SCL_LCNT															

Bit	Name	W/R	Description
31:16	保留	-	

15:0	IC_FS_SCL_LCNT	W/R	I2C SCL 低电平周期计数器 I2C 工作在快速（fast speed）模式下，该寄存器值决定 SCL 低电平保持时间，寄存器最小值为 8。

15.4.10I2C使能寄存器（IC_ENABLE）

- **Name: I2C Enable Register**
- **Size: 32 bits**
- **Address Offset: 0x6C**
- **Read/Write Access: Read/Write**

31	30	29	28	27	26	25	24
保留							
23	22	21	20	19	18	17	16
保留							
15	14	13	12	11	10	9	8
保留							
7	6	5	4	3	2	1	0
保留						ABORT	ENABLE

Bit	Name	W/R	Description
31:2	保留	-	
1	ABORT	W/R	软件终止传输 0-ABORT 为开始或已经完成 1-正在进行 ABORT 操作 I2C 工作在主模式中，通过软件终止 I2C 传输。 只有当 I2C ENABLE 已经被置"1"后才能进行 ABORT 操作，否则忽略 ABORT 操作。 ABORT 位不能通过软件清"0"，由硬件完成 ABORT 操作后自动清理。在 ABORT 操作过程中，硬件会发出 STOP 信号，清空 TX FIFO，并置 ABORT 中断位为"1"。
0	ENABLE	W/R	I2C 使能位 0-I2C 关闭（TX 和 RX FIFO 均处于清空状态） 1-I2C 打开 I2C 关闭操作将会伴随以下状态： 1、TX FIFO 和 RX FIFO 被清空 2、IC_INTR_STAT 会保持直到 I2C 进入 IDLE 状态

15.4.11I2C状态寄存器（IC_STATUS）

- **Name: I2C Status Register**
- **Size: 32 bits**
- **Address Offset: 0x70**
- **Read/Write Access: Read**

该寄存器中bit位置"1"不会引起中断请求。

当I2C关闭后：

- Bits 1 and 2 are set to 1
- Bits 3 and 4 are set to 0

当I2C关闭后并处于IDLE状态：

- Bits 5 and 6 are set to 0

31	30	29	28	27	26	25	24
保留							
23	22	21	20	19	18	17	16
保留							
15	14	13	12	11	10	9	8
保留							
7	6	5	4	3	2	1	0
保留	SLV_ACTIVITY	MST_ACTIVITY	RFF	RFNE	TFE	TFNF	ACTIVITY

Bit	Name	W/R	Description
31:7	保留	-	
6	SLV_ACTIVITY	R	I2C 从模式最终状态机（Slave Finite State Machine）（FSM） 0-从模式 FSM 状态为非 Activity 状态 1-从模式 FSM 状态为 Activity 状态 Activity 状态即 I2C FSM 为非 IDLE 状态。
5	MST_ACTIVITY	R	I2C 主模式最终状态机（Master Finite State Machine）（FSM） 0-主模式 FSM 状态为非 Activity 状态 1-主模式 FSM 状态为 Activity 状态 Activity 状态即 I2C FSM 为非 IDLE 状态。
4	RFF	R	接收 FIFO 满 0-接收 FIFO 未满 1-接收 FIFO 满 该位是否置位与不受 IC_RX_TL 影响
3	RFNE	R	接收 FIFO 非空 0-接收 FIFO 空（FIFO 中没有数据） 1-接收 FIFO 非空（FIFO 中有 1 个及以上数据） 该位是否置位与不受 IC_RX_TL 影响
2	TFE	R	发送 FIFO 空 0-发送 FIFO 非空（FIFO 中有 1 个及以上数据） 1-发送 FIFO 空（FIFO 中没有数据） 该位是否置位与不受 IC_TX_TL 影响
1	TFNF	R	发送 FIFO 未满 0-发送 FIFO 已满 1-发送 FIFO 未满 该位是否置位与不受 IC_TX_TL 影响
0	ACTIVITY	R	I2C 最终状态机（Slave Finite State Machine）（FSM） 0-非 Activity 状态 1-Activity 状态 Activity 状态即 I2C FSM 为非 IDLE 状态。

15.4.12I2C发送FIFO数据量寄存器（IC_TXFLR）

- **Name: I2C Transmit FIFO Level Register**
- **Size: 32 bits**
- **Address Offset: 0x74**
- **Read/Write Access: Read**

I2C发送FIFO中有效数据量，在以下情况被清“0”：
- 关闭（disable）I2C
- 传输终止（transmit abort）

当向FIFO中写入数据时寄存器值增加，当I2C从FIFO中取数据时寄存器值减小。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留												TXFLR			

Bit	Name	W/R	Description
31:4	保留	-	
3:0	TXFLR	R	发送 FIFO 数据量 发送 FIFO 中包含的有效数据数

15.4.13I2C接收FIFO数据量寄存器（IC_RXFLR）

- **Name: I2C ReceiveFIFO Level Register**
- **Size: 32bits**
- **Address Offset: 0x78**
- **Read/Write Access: Read**

I2C接收FIFO中有效数据量，在以下情况被清“0”：
- 关闭（disable）I2C
- 传输终止（transmit abort）

当I2C接收数据到FIFO中时寄存器值增加，当用户从FIFO中取数据时寄存器值减小。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留												RXFLR			

Bit	Name	W/R	Description
31:4	保留	-	
3:0	RXFLR	R	接收 FIFO 数据量 接收 FIFO 中包含的有效数据数

15.4.14I2C SDA HOLD时间寄存器（IC_SDA_HOLD）

- **Name: SDA Hold Time Length Register**
- **Size: 32bits**
- **Address Offset: 0x7C**
- **Read/Write Access: Read/Write**

寄存器值用于控制SCL下降沿产生后SDA需要保持的时间，时间以PCLK周期为单位。I2C在主模式下最小值为1个PCLK周期，在从模式下最小是为7个PCLK周期。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IC_SDA_HOLD															

Bit	Name	W/R	Description
31:16	保留	-	
15:0	IC_SDA_HOLD	W/R	设定 SDA 保持时间，单位时间为 PCLK 周期

15.4.15I2C发送终止源寄存器（IC_TX_ABRT_SOURCE）

- **Name: I2C Transmit Abort Source Register**
- **Size:32bits**
- **Address Offset: 0x80**
- **Read/Write Access: Read**

除寄存器第9位（TX_ABRT），其他位均可以通过对IC_CLR_TX_ABRT或IC_CLR_INTR读操作进行清除。

清除寄存器第9位（TX_ABRT）需满足以下情况：

- RESTART 必须有效（IC_CON[5]=1）
- SPECIAL（IC_TAR[11]）或 GC_OR_START（IC_TAR[10]）必须被清“0”。

满足以上情况后即可通过对IC_CLR_TX_ABRT或IC_CLR_INTR读操作进行清除，如果不满足，清除后TX_ABRT位自动置位。

31	30	29	28	27	26	25	24
TX_FLUSH_CNT							
23	22	21	20	19	18	17	16
保留							ABRT_USER_ABRT
15	14	13	12	11	10	9	8
ABRT_SLVRD_INTX	ABRT_SLV_ARBLOST	ABRT_SLVFLUSH_TXFIFO	ARB_LOST	ABRT_MASTER_DIS	ABRT_10B_RD_NORSTRT	ABRT_SBYTE_NORSTRT	ABRT_HS_NORSTRT
7	6	5	4	3	2	1	0
ABRT_SBYTE_ACKDET	ABRT_HS_ACKDET	ABRT_GCALL_READ	ABRT_GCALL_NOACK	ABRT_TXDATA_NOACK	ABRT_10ADDR2_NOACK	ABRT_10ADDR1_NOACK	ABRT_7B_ADDR_NOACK

Bit	Name	W/R	Mode	Description
31:24	TX_FLUSH_CNT	R	主模式	用于记录在发生 TX_ABRT 事件后 TXFLR 中的值
23:17	保留	-	-	
16	ABRT_USER_ABRT	R	主模式	用户主动终止传输
15	ABRT_SLVRD_INTX	R	从模式	在 I2C 作为从设备时，处理器响应 1 个远程主设备数据读请求，用户对 IC_DATA_CMD 寄存器 CMD (bit 8)写 1
14	ABRT_SLV_ARBLOST	R	从模式	在 I2C 作为从设备向主设备发送

				数据过程中失去总线。
13	ABRT_SLVFLUSH_TXFIFO	R	从模式	在I2C作为从设备时接收到外部主机读请求命令，若此时发送FIFO中有数据则该位置“1”
12	ARB_LOST	R	主/从模式	在I2C作为主设备时I2C总线仲裁失败 在I2C作为从设备时从发送器总线仲裁失败
11	ABRT_MASTER_DIS	R	主/从模式	在I2C主模式关闭的情况下，用户尝试I2C主模式操作
10	ABRT_10B_RD_NORSTRT	R	主模式	在IC_RESTART_EN使能关闭（IC_CON[5] = 0）同时主设备发送10地址模式读命令
9	ABRT_SBYTE_NORSTRT	R	主模式	在IC_RESTART_EN使能关闭（IC_CON[5] = 0）同时用户尝试发送START BYTE
8	ABRT_HS_NORSTRT	R	主模式	在IC_RESTART_EN使能关闭（IC_CON[5] = 0）同时用户尝试使用主模式在高速模式下传输数据
7	ABRT_SBYTE_ACKDET	R	主模式	I2C工作在主模式下发送START BYTE，接收到ACK
6	ABRT_HS_ACKDET	R	主模式	I2C工作在高速主模式下发送START BYTE，接收到ACK
5	ABRT_GCALL_READ	R	主模式	I2C工作在主模式发送地址呼叫（General Call）而用户在发送地址呼叫完成后IC_DATA_CMD[9]置“1”尝试发送读命令
4	ABRT_GCALL_NOACK	R	主模式	I2C工作在主模式发送地址呼叫（General Call），总线上没有从设备响应
3	ABRT_TXDATA_NOACK	R	主模式	该bit值在主模式中使用。 I2C工作在主模式中，可以收到对方地址ACK，但在发送数据，没有收到ACK响应。
2	ABRT_10ADDR2_NOACK	R	主模式	I2C工作在10地址模式，第2字节地址没有从机ACK响应
1	ABRT_10ADDR1_NOACK	R	主模式	I2C工作在10地址模式，第1字节地址没有从机ACK响应
0	ABRT_7B_ADDR_NOACK	R	主模式	I2C工作在7地址模式，发送地址命令字节，没有从机ACK响应

15.4.16I2C从模式数据NACK应答寄存器（IC_SLV_DATA_NACK_ONLY）

- **Name: Generate Slave Data NACK Register**
- **Size:32bits**
- **Address Offset: 0x84**
- **Read/Write Access: Read/Write**

I2C工作在从模式下，用于在数据传输过程中产生NACK信号。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留															
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
保留															NACK

Bit	Name	W/R	Description
31:1	保留	-	
0	NACK	W/R	I2C 工作在从模式下，用于在数据传输过程中产生 NACK 信号。 1-在字节数据接收完成后产生 NACK 0-正常模式产生 NACK/ACK

15.4.17I2C SDA SETUP时间设定寄存器（IC_SDA_SETUP）

- **Name: SDA Setup Register**
- **Size: 32 bits**
- **Address Offset: 0x94**
- **Read/Write Access: Read/Write**

寄存器值用于控制SCL上升沿产生后与SDA有效间的时间延时（详见I2C Bus Specification中tSU:DAT），时间以PCLK周期为单位。该寄存器编程值需大于或等于2。

该寄存器仅在IC_ENABLE[0] = 0时操作有效。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留															
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
保留								IC_SDA_SETUP							

Bit	Name	W/R	Description
31:8	保留	-	
7:0	IC_SDA_SETUP	W/R	SDA SETUP 设定 寄存器值用于控制 SCL 上升沿产生后与 SDA 有效间的时间延时（详见 I2C Bus Specification 中 tSU:DAT），时间以 PCLK 周期为单位。该寄存器编程值需大于或等于 2。

15.4.18I2C地址呼叫响应寄存器（IC_ACK_GENERAL_CALL）

- **Name: ACK General Call Register**
- **Size: 32 bits**
- **Address Offset: 0x98**
- **Read/Write Access: Read/Write**

寄存器控制I2C使用NACK或ACK响应地址呼叫（General Call）。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留															
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**

保留	ACK_GEN_CALL

Bit	Name	W/R	Description
31:1	保留	-	
0	ACK_GEN_CALL	W/R	地址呼叫（General Call）响应 0-不响应地址呼叫 1-响应地址呼叫

15.4.19I2C使能状态寄存器（IC_ENABLE_STATUS）

- **Name: I2C Enable Status Register**
- **Size: 32 bits**
- **Address Offset: 0x 9C**
- **Read/Write Access: Read**

寄存器用于反映IC_ENABLE[0]由1置0后I2C的状态。
当IC_ENABLE[0] = 1：SLV_RX_DATA_LOST和SLV_DISABLED_WHILE_BUSY强制置“0”，IC_EN强制置“1”。
当IC_ENABLE[0] = 0：直到IC_EN由硬件置“1”后，SLV_RX_DATA_LOST和SLV_DISABLED_WHILE_BUSY的值有效。

31	30	29	28	27	26	25	24
保留							
23	22	21	20	19	18	17	16
保留							
15	14	13	12	11	10	9	8
保留							
7	6	5	4	3	2	1	0
保留					SLV_RX_DATA_LOST	SLV_DISABLED_WHILE_BUSY	IC_EN

Bit	Name	W/R	Description
31:3	保留	-	
2	SLV_RX_DATA_LOST	W/R	从模式下数据丢失 当 I2C 在从模式下 IC_ENABLE[0]由“1”置“0”，接收被终止且接收 FIFO 中至少有 1 个有效数据的情况下，该 bit 位置“1”。
1	SLV_DISABLED_WHILE_BUSY	W/R	I2C 从模式忙状态下关闭 I2C 工作在从模式下且处于忙状态时 IC_ENABLE[0]由“1”置“0”，该位置“1”。
0	IC_EN	W/R	I2C 使能状态 0-I2C 已关闭 1-I2C 已打开

15.4.20I2C标准/快速模式毛刺长度寄存器（IC_FS_SPKLEN）

- **Name: I2C SS and FS Spike Suppression Limit Register**
- **Size:32 bits**
- **Address Offset: 0x a0**

- **Read/Write Access: Read/Write**

毛刺过滤参数I2C Bus Specification中tSP相关介绍。
该寄存器仅在IC_ENABLE[0] = 0时操作有效。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留								IC_FS_SPKLEN							

Bit	Name	W/R	Description
31:8	保留	-	
7:0	IC_FS_SPKLEN	W/R	SDA/SCL 中大于设定值的毛刺将被 I2C 逻辑单元过滤 最小值为 1

16 SPI接口

16.1 SPI简介

串行外设接口(SPI)允许芯片与外部设备以半/全双工、同步、串行方式通信。此接口可以被配置成主模式，并为外部从设备提供通信时钟(SCK)。接口还能以多主配置方式工作。

16.2 SPI主要特点

- SPI时钟由PCLK提供，即SPI_CLK = PCLK
- 支持协议Motorola Serial Peripheral Interface (SPI)
- 支持协议Texas Instruments Serial Protocol (SSP)
- 支持协议National Semiconductor Microwire
- 包含硬件实现收发FIFO，深度为16
- 独立硬件收发FIFO，可配收发FIFO中断阈值
- SPI0支持主或从操作（主/从地址不同）
- 从模式支持CS拉低时持续接收
- 4到16位数据帧格式选择
- 支持全双工、半双工模式
- 多种收发、错误中断检测
- DMA支持

16.3 SPI功能描述

16.3.1 SPI外设时钟及要求

SPI时钟由PCLK提供，即SPI_CLK = PCLK

SPI_CLK：SPI接入时钟频率
SPI_M_CLK：SPI主模式总线时钟频率
SPI_S_CLK：SPI从模式总线时钟频率

理论时钟要求：
SPI_CLK >= 2 x SPI_M_CLK
SPI_CLK >= 10 x SPI_S_CLK

16.3.2 接收/发送FIFO

SPI外设包含2个独立的深度为16的接收和发送FIFO。

CPU对寄存器DR写操作，数据写入发送FIFO，CPU对寄存器DR读操作，数据由接收FIFO中取出。

接收和发送FIFO有独立的中断阈值设定，当数据符合设定阈值时SPI向CPU发出中断请求。

接收和发送FIFO有独立的DMA阈值设定，当数据符合设定阈值时SPI发出相应DMA请求。

设定的阈值需结合DMA Burst（MSIZE）值进行设定，相见DMA“SRC_MSIZE/DEST_MSIZE参数置设定”章节

16.3.3 中断类型

SPI外设支持以下中断类型：

- Transmit FIFO Empty Interrupt

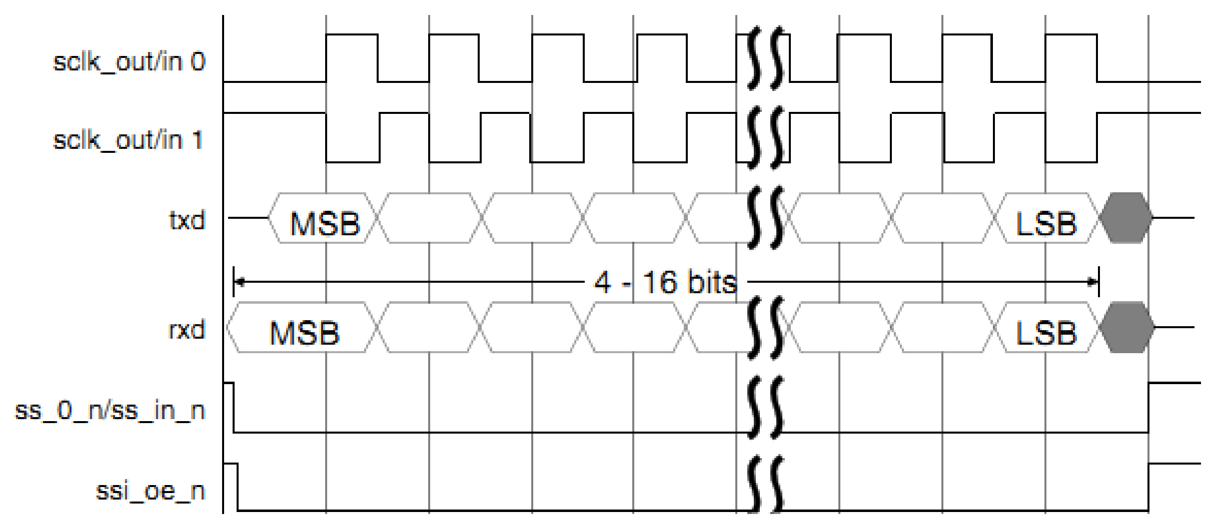
发送 FIFO 空中断，当发送 FIFO 中数据量满足设定阈值时产生中断

- Transmit FIFO Overflow Interrupt
  发送 FIFO 上溢中断，在发送 FIFO 数据已经满的情况下对 DR 进行写操作引起发送 FIFO 上溢中断
- Receive FIFO Full Interrupt
  接收 FIFO 满中断，当接收 FIFO 中数据量满足设定阈值时产生中断
- Receive FIFO Overflow Interrupt
  接收 FIFO 上溢中断，在接收 FIFO 数据已经满的情况下 SPI 外设接收新数据引起接收 FIFO 上溢中断
- Receive FIFO Underflow Interrupt
  接收 FIFO 下溢中断，接收 FIFO 已经为空，对接收 FIFO 进行读操作会触发接收 FIFO 下溢中断
- Multi-Master Contention Interrupt
  多主机总线仲裁中断，SPI 工作在住模式下并占用总先，此时有其他主设备选中当前 SPI 外设并传输数据。
- Combined Interrupt Request
  以上中断类型经过中断屏蔽寄存器后或运算值

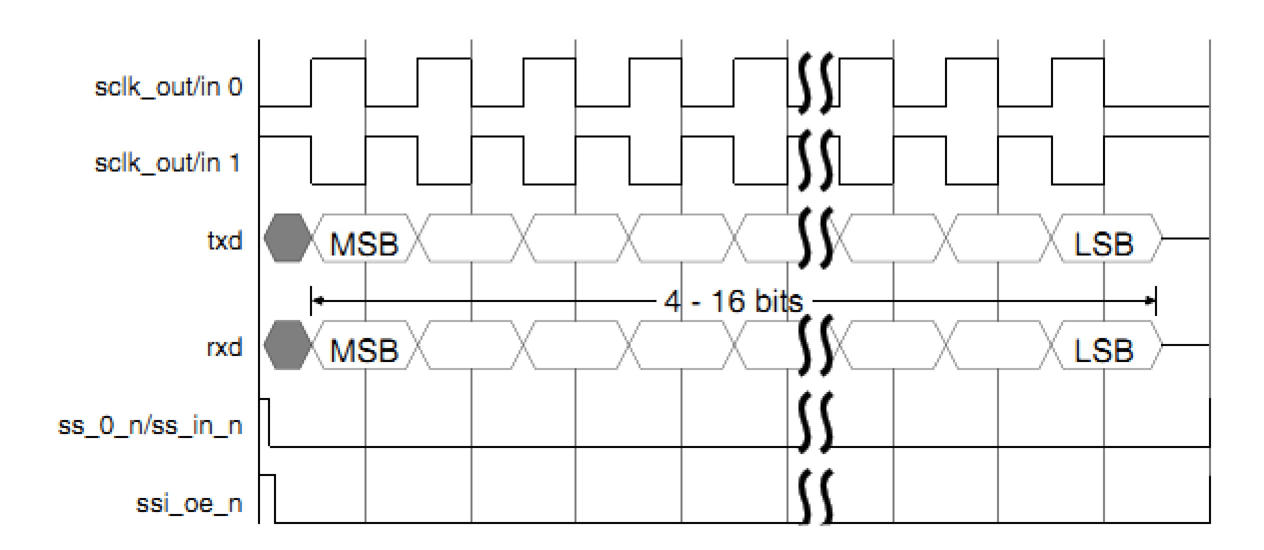
16.3.4 Motorola SPI通行协议

常用Motorola SPI通讯协议支持的四种通讯模式，能够实现全双工通讯。系统上电默认采用模式0工作方式。

SCPH = 0:



SCPH = 1:



sclk_out/ in：总线时钟，out：SPI为主设备输出CLK。in：SPI为从设备输入CLK
sclk_out/ in = 0：（CPHA）=0
sclk_out/ in = 1：（CPHA）=1
ss_0_n/ss_in_n：片选信号，s_0_n：SPI为主设备时输出片选。s_in_n：SPI为主设备时输入片选
ss_oe_n：SPI为从模式时输出使能选项。

SPI协议规定的4中通讯格式说明如下：

- 模式0：时钟极性（CPOL）=0，时钟相位（CPHA）=0，该模式下串行同步时钟的空闲状态为低电平，芯片将在串行同步时钟的第一个跳变沿（上升沿）采样，芯片默认为该模式；
- 模式1：时钟极性（CPOL）=0，时钟相位（CPHA）=1，该模式下串行同步时钟的空闲状态为低电平，芯片将在串行同步时钟的第二个跳变沿（下降沿）采样；
- 模式2：时钟极性（CPOL）=1，时钟相位（CPHA）=0，该模式下串行同步时钟的空闲状态为高电平，芯片将在串行同步时钟的第一个跳变沿（下降沿）采样；
- 模式3：时钟极性（CPOL）=1，时钟相位（CPHA）=1，该模式下串行同步时钟的空闲状态为高电平，芯片将在串行同步时钟的第二个跳变沿（上升沿）采样。

16.3.5 SPI主/从模式选择

SPI0包括2组寄存器组，分别用于实现主模式（SPIM0）和从模式（SPIS0），2组寄存器组结构相同，地址不同。SPI外设工作模式使用SYSCTRL->PHER_CTRL寄存器相应位切换。

当工作在主模式下，SPI相应初始化及数据收发操作由SPIMx完成。当工作在从模式下，SPI相应初始化及数据接收操作有SPISx完成

使能SPI0时钟门控并使其处在主模式下，具体流程如下：

```
// CG_CTRL为时钟门控寄存器详见系统控制（SYSCTL）章节
CG_CTRL |= 1<<8
// PHER_CTRL为外设控制寄存器详见系统控制（SYSCTL）章节
PHER_CTRL &= ~(1<<24)
```

使能SPI0时钟门控并使其处在从模式下，具体流程如下：

```
// CG_CTRL为时钟门控寄存器详见系统控制（SYSCTL）章节
CG_CTRL |= 1<<8
// PHER_CTRL为外设控制寄存器详见系统控制（SYSCTL）章节
PHER_CTRL |= 1<<24
```

16.3.6 SPI主模式配置

主模式使用SPIMx寄存器组进行配置
配置流程如下：

1、 配置CG_CTRL使SPI处于主模式下
2、 配置SPIMx->SSIENR = 0，SPI使能关闭
3、 配置SPIMx-> IMR= 0，关中断
4、 配置SPIMx-> CTRLR0，配置传输模式，帧模式，时钟极性，时钟相位，及通讯数据宽度
5、 配置SPIMx-> BAUDR，配置需要的波特率
6、 配置SPIMx-> SER，初始化片选信号
7、 配置SPIMx-> RXFTLR和SPIMx-> TXFTLR，配置收发FIFO中断触发阈值
8、 配置SPIMx-> IMR，开启相关中断
9、 配置SPIMx->SSIENR = 1，SPI使能打开

16.3.7 SPI主模式数据收发

主模式使用SPIMx寄存器组进行数据发送/接收
数据发送/接收流程：

1、 判断发送 FIFO 是否已满。
2、 如果发送 FIFO 未满，向 SPIMx->DR 寄存器中写数据。数据将发送到对应片选从设备。
3、 如果 SPIMx-> SER 没有使能任何从设备，则在 SPIMx-> SER 是使能后数据被发送。
4、 当发生发送 FIFO 空中断时，向 SPIMx-> DR 写数据到发送 FIFO 中。当发生接收 FIFO 满时，读 SPIMx-> DR 由发送 FIFO 读取数据。
5、 数据传输完成，SPI 回到非忙状态

16.3.8 SPI从模式配置

从模式使用SPISx寄存器组进行配置
配置流程如下：

1、 配置 CG_CTRL 使 SPI 处于从模式下
2、 配置 SPISx->SSIENR = 0，SPI 使能关闭
3、 配置 SPISx-> IMR= 0，关中断
4、 配置 SPISx-> CTRLR0，配置传输模式，帧模式，时钟极性，时钟相位，从输出使能
5、 配置 SPISx-> RXFTLR 和 SPISx-> TXFTLR，配置收发 FIFO 中断触发阈值
6、 配置 SPISx-> IMR，开启相关中断
7、 配置 SPISx->SSIENR = 1，SPI 使能打开

16.3.9 SPI从模式数据收发

从模式使用SPISx寄存器组进行数据发送/接收
数据发送/接收流程：

1、 当 SPI 片选有效后，数据开始传输。
2、 当发生发送 FIFO 空中断时，向 SPISx-> DR 写数据到发送 FIFO 中。当发生接收 FIFO 满时，读 SPISx-> DR 由发送 FIFO 读取数据。
3、 数据传输完成，SPI 回到非忙状态

16.3.10DMA操作

SPI外设支持DMA数据操作，以减少CPU使用。

DMACR寄存器：

该寄存器用于使能SPI收发DMA功能，分别有2bit控制位操作。

DMATDLR/ DMARDLR寄存器：

DMATDLR/ DMARDLR寄存器用于控制SPI对DMA产生请求条件。

当发送FIFO中数据满足DMATDLR设定值，则触发SPI对DMA请求，DMA相应请求后，根据对应通道配置向发送FIFO中1次性写入Burst（MSIZE）数据量的数据。

当接收FIFO中数据满足DMARDLR设定值，则触发SPI对DMA请求，DMA相应请求后，根据对应通道配置由发送FIFO中1次性取出Burst（MSIZE）数据量的数据。

具体设定可参考DMA中“SRC_MSIZE/DEST_MSIZE参数置设定”章节。

16.4 SPI寄存器描述

16.4.1 地址映射表

SPIx（x=0...2）基地址列表

地址范围	基地址	外设	总线
0x4001_A000-0x4001_AFFF	0x4001_A000	SPIM0	APB0
0x4001_B000-0x4001_BFFF	0x4001_B000	SPIS0	
0x4001_8000-0x4001_8FFF	0x4001_8000	SPIM1	
0x4001_9000-0x4001_9FFF	0x4001_9000	SPIM2	

表 16-1 SPI寄存器表

偏移地址	寄存器名称	宽度（bit）	复位值
0x00	CTRLR0	32	0x00000007
0x04	CTRLR1	32	0x00000000
0x08	SSIENR	32	0x00000000
0x0C	MWCR	32	0x00000000
0x10	SER	32	0x00000000
0x14	BAUDR	32	0x00000000
0x18	TXFTLR	32	0x00000000
0x1C	RXFTLR	32	0x00000000
0x20	TXFLR	32	0x00000000
0x24	RXFLR	32	0x00000000
0x28	SR	32	0x00000006
0x2C	IMR	32	0x0000003F
0x30	ISR	32	0x00000000
0x34	RISR	32	0x00000000
0x38	TXOICR	32	0x00000000
0x3C	RXOICR	32	0x00000000
0x40	RXUICR	32	0x00000000
0x44	MSTICR	32	0x00000000
0x48	ICR	32	0x00000000
0x4C	DMACR	32	0x00000000
0x50	DMATDLR	32	0x00000000
0x54	DMARDLR	32	0x00000000
0x58	预留	32	0xFFFFFFFF
0x5C	预留	32	0x3332322A
0x60	DR	32	0x00000000
0x64	DR	32	0x00000000
0x68	DR	32	0x00000000
0x6C	DR	32	0x00000000

0x70	DR	32	0x00000000
0x74	DR	32	0x00000000
0x78	DR	32	0x00000000
0x7C	DR	32	0x00000000
0x80	DR	32	0x00000000
0x84	DR	32	0x00000000
0x88	DR	32	0x00000000
0x8C	DR	32	0x00000000
0x90	DR	32	0x00000000
0x94	DR	32	0x00000000
0x98	DR	32	0x00000000
0x9C	DR	32	0x00000000
0xA0	DR	32	0x00000000
0xA4	DR	32	0x00000000
0xA8	DR	32	0x00000000
0xAC	DR	32	0x00000000
0xB0	DR	32	0x00000000
0xB4	DR	32	0x00000000
0xB8	DR	32	0x00000000
0xBC	DR	32	0x00000000
0xC0	DR	32	0x00000000
0xC4	DR	32	0x00000000
0xC8	DR	32	0x00000000
0xCC	DR	32	0x00000000
0xD0	DR	32	0x00000000
0xD4	DR	32	0x00000000
0xD8	DR	32	0x00000000
0xDC	DR	32	0x00000000
0xE0	DR	32	0x00000000
0xE4	DR	32	0x00000000
0xE8	DR	32	0x00000000
0xEC	DR	32	0x00000000
0xF0	RX_SAMPLE_DLY	32	0x00000000

16.4.2 控制寄存器0（CTRLR0）

- **Name: Control Register 0**
- **Size:** 32 bits
- **Address Offset: 0x0**
- **Read/write access:** read/write

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
预留															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CFS				SRL	SLV_OE	TMOD		SCPOL	SCPH	FRF		DFS			

Bit	Name	W/R	Description
31:16	预留	--	预留
15:12	CFS	W/R	控制帧大小 Microwire 帧格式，设定值参见下表：

<table>
<tr><td></td><td></td><td></td><td><table>
<tr><td>0000</td><td>1-bit control word</td></tr>
<tr><td>0001</td><td>2-bit control word</td></tr>
<tr><td>0010</td><td>3-bit control word</td></tr>
<tr><td>0011</td><td>4-bit control word</td></tr>
<tr><td>0100</td><td>5-bit control word</td></tr>
<tr><td>0101</td><td>6-bit control word</td></tr>
<tr><td>0110</td><td>7-bit control word</td></tr>
<tr><td>0111</td><td>8-bit control word</td></tr>
<tr><td>1000</td><td>9-bit control word</td></tr>
<tr><td>1001</td><td>10-bit control word</td></tr>
<tr><td>1010</td><td>11-bit control word</td></tr>
<tr><td>1011</td><td>12-bit control word</td></tr>
<tr><td>1100</td><td>13-bit control word</td></tr>
<tr><td>1101</td><td>14-bit control word</td></tr>
<tr><td>1110</td><td>15-bit control word</td></tr>
<tr><td>1111</td><td>16-bit control word</td></tr>
</table></td></tr>
<tr><td>11</td><td>SRL</td><td>W/R</td><td>移位寄存器环
该位仅用于测试，该位置“1”则输出移位寄存器自动连接到输入移位寄存器上。
0-正常操作模式
1-测试操作模式</td></tr>
<tr><td>10</td><td>SLV_OE</td><td>W/R</td><td>从输出使能
该位仅用于模块工作在从模式，当工作在主模式是对该位操作无效
0-从发送打开
1-从发送关闭</td></tr>
<tr><td>9:8</td><td>TMOD</td><td>W/R</td><td>传输模式
00-发送和接收模式
01-只发送
10-只接收
11-EEPROM 模式</td></tr>
<tr><td>7</td><td>SCPOL</td><td>W/R</td><td>时钟极性
0-空闲状态时，SCK 保持低电平；
1-空闲状态时，SCK 保持高电平。</td></tr>
<tr><td>6</td><td>SCPH</td><td>W/R</td><td>时钟相位
0-数据采样从第一个时钟边沿开始；
1-数据采样从第二个时钟边沿开始。</td></tr>
<tr><td>5:4</td><td>FRF</td><td>W/R</td><td>协议帧格式
00-Motorola SPI
01-Texas Instruments SSP
10-National Semiconductors Microwire
11-Reserved</td></tr>
<tr><td>3:0</td><td>DFS</td><td>W/R</td><td>数据帧大小
设定值参考下表：<table>
<tr><td>0000</td><td>预留</td></tr>
<tr><td>0001</td><td>预留</td></tr>
<tr><td>0010</td><td>预留</td></tr>
<tr><td>0011</td><td>4-bit data size</td></tr>
<tr><td>0100</td><td>5-bit data size</td></tr>
<tr><td>0101</td><td>6-bit data size</td></tr>
<tr><td>0110</td><td>7-bit data size</td></tr>
<tr><td>0111</td><td>8-bit data size</td></tr>
</table></td></tr>
</table>

			1000	9-bit data size
			1001	10-bit data size
			1010	11-bit data size
			1011	12-bit data size
			1100	13-bit data size
			1101	14-bit data size
			1110	15-bit data size
			1111	16-bit data size

16.4.3 控制寄存器1（CTRLR1）

- **Name: Control Register 1**
- **Size:** 32 bits
- **Address Offset: 0x04**
- **Read/write access:** read/write

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
预留															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NDF															

Bit	Name	R/W	Description
31:16	预留	--	预留
15:0	NDF	R/W	数据帧数 在 TMOD = 10 或 TMOD = 11 后，该寄存器用于设定连续数据帧接收数量。

16.4.4 使能寄存器（SSIENR）

- **Name: SSI Enable Register**
- **Size:**32bits
- **Address Offset: 0x08**
- **Read/write access:** read/write

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
预留															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
预留															SSIENR

Bit	Name	W/R	Description
31:1	预留	--	预留
0	SSIENR	W/R	使能寄存器 0-SPI 关闭 1-SPI 打开

16.4.5 Microwire控制寄存器（MWCR）

- **Name: Microwire Control Register**
- **Size:** 32bits
- **Address Offset: 0x0C**

- **Read/write access:** read/write

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
预留															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
预留													MHS	MDD	MWMOD

Bit	Name	W/R	Description
31:3	预留	-	预留
2	MHS	W/R	Microwire 握手 该位仅在 SPI 被配置为主模式时有效，从模式时该位忽略。 该位用于使能 Microwire “busy/ready”握手接口。当使能打开，SPI 发送完最后 1bit 数据或控制命令后，将检测远程从设备 ready 状态。
1	MDD	W/R	Microwire 方向控制位 0-接收外部设备数据 1-发送数据到外部设备
0	MWMOD	W/R	Microwire 传输模式 0-非连续传输 1-连续传输

16.4.6 从设备选择寄存器（SER）

- **Name: Slave Enable Register**
- **Size:** 32bits
- **Address Offset: 0x10**
- **Read/write access:** read/write

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
预留															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
预留															s0

Bit	Name	W/R	Description
31:1	预留	-	预留
0	s0	W/R	从设备选择寄存器 该寄存器仅在 SPI 为主模式状态下有效，从模式状态下忽略。 0-从设备无效 1-从设备有效

16.4.7 波特率寄存器（BAUDR）

- **Name: Baud Rate Select**
- **Size:** 32bits
- **Address Offset: 0x14**
- **Read/write access:** read/write

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
预留															
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
SCKDV															

Bit	Name	W/R	Description
31:16	预留	-	预留
15:0	SCKDV	W/R	波特率寄存器 波特率 = PCLK / SCKDV SCKDV 范围 2 到 65534（只能为偶数） 当 SPI 使能打开后，对该寄存器操作有效。

16.4.8 发送FIFO阈值寄存器（TXFTLR）

- **Name: Transmit FIFO Threshold Level**
- **Size: 32bits**
- **Address Offset: 0x18**
- **Read/Write Access: Read/Write**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
预留															
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
预留												TXFTLR			

<table>
<tr><th>Bit</th><th>Name</th><th>W/R</th><th>Description</th></tr>
<tr><td>31:4</td><td>预留</td><td>-</td><td>预留</td></tr>
<tr><td>3:0</td><td>TXFTLR</td><td>W/R</td><td>发送 FIFO 满中断阈值

设定值参考下表：
<table>
<tr><td>0</td><td>发送 FIFO 中数据数量为 0 触发中断</td></tr>
<tr><td>1</td><td>发送 FIFO 中数据少于 1 个触发中断</td></tr>
<tr><td>2</td><td>发送 FIFO 中数据少于 2 个触发中断</td></tr>
<tr><td>3</td><td>发送 FIFO 中数据少于 3 个触发中断</td></tr>
<tr><td>4</td><td>发送 FIFO 中数据少于 4 个触发中断</td></tr>
<tr><td>5</td><td>发送 FIFO 中数据少于 5 个触发中断</td></tr>
<tr><td>6</td><td>发送 FIFO 中数据少于 6 个触发中断</td></tr>
<tr><td>7</td><td>发送 FIFO 中数据少于 7 个触发中断</td></tr>
<tr><td>8</td><td>发送 FIFO 中数据少于 8 个触发中断</td></tr>
<tr><td>9</td><td>发送 FIFO 中数据少于 9 个触发中断</td></tr>
<tr><td>10</td><td>发送 FIFO 中数据少于 10 个触发中断</td></tr>
<tr><td>11</td><td>发送 FIFO 中数据少于 11 个触发中断</td></tr>
<tr><td>12</td><td>发送 FIFO 中数据少于 12 个触发中断</td></tr>
<tr><td>13</td><td>发送 FIFO 中数据少于 13 个触发中断</td></tr>
<tr><td>14</td><td>发送 FIFO 中数据少于 14 个触发中断</td></tr>
<tr><td>15</td><td>发送 FIFO 中数据少于 15 个触发中断</td></tr>
</table>
</td></tr>
</table>

16.4.9 接收FIFO阈值寄存器（RXFTLR）

- **Name: Receive FIFO Threshold Level**
- **Size: 32bits**

- **Address Offset: 0x1C**
- **Read/Write Access: Read/Write**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
预留															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
预留												RXFTLR			

Bit	Name	W/R	Description
31:4	预留	-	预留
3:0	RXFTLR	W/R	接收 FIFO 空中断阈值 设定值参考下表： 0：接收 FIFO 中数据 1 个及以上触发中断 1：接收 FIFO 中数据 2 个及以上触发中断 2：接收 FIFO 中数据 3 个及以上触发中断 3：接收 FIFO 中数据 4 个及以上触发中断 4：接收 FIFO 中数据 5 个及以上触发中断 5：接收 FIFO 中数据 6 个及以上触发中断 6：接收 FIFO 中数据 7 个及以上触发中断 7：接收 FIFO 中数据 8 个及以上触发中断 8：接收 FIFO 中数据 9 个及以上触发中断 9：接收 FIFO 中数据 10 个及以上触发中断 10：接收 FIFO 中数据 11 个及以上触发中断 11：接收 FIFO 中数据 12 个及以上触发中断 12：接收 FIFO 中数据 13 个及以上触发中断 13：接收 FIFO 中数据 14 个及以上触发中断 14：接收 FIFO 中数据 15 个及以上触发中断 15：接收 FIFO 中数据 16 个触发中断

16.4.10发送FIFO数据量寄存器（TXFLR）

- **Name: Transmit FIFO Level Register**
- **Size: 32bits**
- **Address Offset: 0x20**
- **Read/Write Access: Read**

当向FIFO中写入数据时寄存器值增加，当SPI从FIFO中取数据时寄存器值减小。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
预留															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
预留												TXFLR			

Bit	Name	W/R	Description
31:4	预留	-	预留
3:0	TXFLR	R	发送 FIFO 数据量 发送 FIFO 中包含的有效数据数

16.4.11接收FIFO数据量寄存器（RXFLR）

- **Name: Receive FIFO Level Register**
- **Size: 32bits**
- **Address Offset: 0x20**
- **Read/Write Access: Read**

当向FIFO中写入数据时寄存器值增加，当SPI从FIFO中取数据时寄存器值减小。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
预留															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
预留												RXFLR			

Bit	Name	W/R	Description
31:4	预留	-	预留
3:0	RXFLR	R	接收 FIFO 数据量 接收 FIFO 中包含的有效数据数

16.4.12状态寄存器（SR）

- **Name: Status Register**
- **Size: 32 bits**
- **Address Offset: 0x28**
- **Read/Write Access: Read**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
预留															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
预留									DCOL	TXE	RFF	RFNE	TFE	TFNF	BUSY

Bit	Name	W/R	Description
31:7	预留	-	预留
6	DCOL	R	数据碰撞错误 0-无错误 1-传输数据碰撞错误 读寄存器清除该位。
5	TXE	R	传输错误 0-无错误 1-传输错误 读寄存器清除该位。
4	RFF	R	接收 FIFO 满 0-接收 FIFO 未满 1-接收 FIFO 满 当 FIFO 未满时，由硬件自动清“0”
3	RFNE	R	接收 FIFO 未空 0-接收 FIFO 空 1-接收 FIFO 未空

			通过软件读 FIFO 清“0”
2	TFE	R	发送 FIFO 空 0-发送 FIFO 未空 1-发送 FIFO 空 当 FIFO 未空时，由硬件自动清“0”
1	TFNF	R	发送 FIFO 未满 0-发送 FIFO 满 1-发送 FIFO 未满 当 FIFO 满时，由硬件自动清“0”
0	BUSY	R	忙状态 0-SPI 处于 EDLE 或关闭状态 1-SPI 处于 Activity 传输数据状态

16.4.13中断屏蔽寄存器（IMR）

- **Name: Interrupt Mask Register**
- **Size: 32bits**
- **Address Offset: 0x2C**
- **Read/Write Access: read/write**

用于屏蔽或允许SPI各中断源。

31	30	29	28	27	26	25	24
预留							
23	22	21	20	19	18	17	16
预留							
15	14	13	12	11	10	9	8
预留							
7	6	5	4	3	2	1	0
预留		MSTIM	RXFIM	RXOIM	RXUIM	TXOIM	TXEIM

Bit	Name	W/R	Description
31:6	预留	-	预留
5	MSTIM	W/R	多主机竞争中断屏蔽 0-禁止多主机竞争中断 1-允许多主机竞争中断
4	RXFIM	W/R	接收 FIFO 满中断屏蔽 0-禁止接收 FIFO 满中断 1-允许接收 FIFO 满中断
3	RXOIM	W/R	接收 FIFO 上溢中断屏蔽 0-禁止接收 FIFO 上溢中断 1-允许接收 FIFO 上溢中断
2	RXUIM	W/R	接收 FIFO 下溢中断屏蔽 0-禁止接收 FIFO 下溢中断 1-允许接收 FIFO 下溢中断
1	TXOIM	W/R	发送 FIFO 上溢中断屏蔽 0-禁止发送 FIFO 上溢中断 1-允许发送 FIFO 上溢中断
0	TXEIM	W/R	发送 FIFO 空中断屏蔽

			0-禁止发送 FIFO 空中断 1-允许发送 FIFO 空中断

16.4.14中断状态寄存器（ISR）

- **Name: Interrupt Status Register**
- **Size: 32bits**
- **Address Offset: 0x30**
- **Read/Write Access: read**

31	30	29	28	27	26	25	24
预留							
23	22	21	20	19	18	17	16
预留							
15	14	13	12	11	10	9	8
预留							
7	6	5	4	3	2	1	0
预留		MSTIS	RXFIS	RXOIS	RXUIS	TXOIS	TXEIS

Bit	Name	W/R	Description
31:6	预留	-	预留
5	SSTIS	R	多主机竞争中断状态 0-未产生多主机竞争中断 1-产生多主机竞争中断
4	RXFIS	R	接收 FIFO 满中断状态 0-未产生接收 FIFO 满中断 1-产生接收 FIFO 满中断
3	RXOIS	R	接收 FIFO 上溢中断状态 0-未产生接收 FIFO 上溢中断 1-产生接收 FIFO 上溢中断
2	RXUIS	R	接收 FIFO 下溢中断状态 0-未产生接收 FIFO 下溢中断 1-产生接收 FIFO 下溢中断
1	TXOIS	R	发送 FIFO 上溢中断状态 0-未产生发送 FIFO 上溢中断 1-产生发送 FIFO 上溢中断
0	TXEIS	R	发送 FIFO 空中断状态 0-未产生发送 FIFO 空中断 1-产生发送 FIFO 空中断

16.4.15原中断状态寄存器（ISR）

- **Name: Raw Interrupt Status Register**
- **Size: 32bits**
- **Address Offset: 0x34**
- **Read/Write Access: read**

该寄存器状态值与中断状态寄存器(ISR)中不同在于，该寄存器的值不受中断屏蔽寄存器(IMR)控制。

31	30	29	28	27	26	25	24

预留							
23	22	21	20	19	18	17	16
预留							
15	14	13	12	11	10	9	8
预留							
7	6	5	4	3	2	1	0
预留		MSTIR	RXFIR	RXOIR	RXUIR	TXOIR	TXEIR

Bit	Name	W/R	Description
31:6	预留	-	预留
5	SSTIR	R	多主机竞争中断状态 0-未产生多主机竞争中断 1-产生多主机竞争中断
4	RXFIR	R	接收 FIFO 满中断状态 0-未产生接收 FIFO 满中断 1-产生接收 FIFO 满中断
3	RXOIR	R	接收 FIFO 上溢中断状态 0-未产生接收 FIFO 上溢中断 1-产生接收 FIFO 上溢中断
2	RXUIR	R	接收 FIFO 下溢中断状态 0-未产生接收 FIFO 下溢中断 1-产生接收 FIFO 下溢中断
1	TXOIR	R	发送 FIFO 上溢中断状态 0-未产生发送 FIFO 上溢中断 1-产生发送 FIFO 上溢中断
0	TXEIR	R	发送 FIFO 空中断状态 0-未产生发送 FIFO 空中断 1-产生发送 FIFO 空中断

16.4.16发送FIFO上溢中断清除寄存器（TXOICR）

- **Name: Transmit FIFO Overflow Interrupt Clear Register**
- **Size: 32bits**
- **Address Offset: 0x38**
- **Read/Write Access: read**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
预留															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
预留															TXOICR

Bit	Name	W/R	Description
31:1	预留	-	预留
0	TXOICR	R	发送 FIFO 上溢中断清除 对其读操作清除中断

16.4.17接收FIFO上溢中断清除寄存器（RXOICR）

- **Name: Receive FIFO Overflow Interrupt Clear Register**
- **Size: 32bits**
- **Address Offset: 0x38**

- **Read/Write Access: read**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
预留															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
预留															RXOICR

Bit	Name	W/R	Description
31:1	预留	-	预留
0	RXOICR	R	接收 FIFO 上溢中断清除 对其读操作清除中断

16.4.18接收FIFO下溢中断清除寄存器（RXUICR）

- **Name: Receive FIFO Underflow Interrupt Clear Register**
- **Size: 32bits**
- **Address Offset: 0x40**
- **Read/Write Access: read**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
预留															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
预留															RXUICR

Bit	Name	W/R	Description
31:1	预留	-	预留
0	RXUICR	R	接收 FIFO 下溢中断清除 对其读操作清除中断

16.4.19多主机竞争中断清除寄存器（MSTICR）

- **Name: Multi-Master Interrupt Clear Register**
- **Size: 32bits**
- **Address Offset: 0x44**
- **Read/Write Access: read**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
预留															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
预留															MSTICR

Bit	Name	W/R	Description
31:1	预留	-	预留
0	MSTICR	R	多主机竞争中断清除 对其读操作清除中断

16.4.20全局中断清除寄存器（ICR）

- **Name: Interrupt Clear Register**
- **Size: 32bits**
- **Address Offset: 0x48**

- **Read/Write Access: read**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
预留															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
预留															ICR

Bit	Name	W/R	Description
31:1	预留	-	预留
0	ICR	R	全局中断清除 对其读操作清除以上 4 个中断状态

16.4.21DMA控制寄存器（DMACR）

- **Name: DMA Control Register**
- **Size: 32 bits**
- **Address Offset: 0x4C**
- **Read/Write Access: read/write**

对该寄存器的操作不受SPI使能的影响。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
预留															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
预留														TDMAE	RDMAE

Bit	Name	W/R	Description
31:2	预留	-	预留
1	TDMAE	W/R	发送 DMA 使能 0-发送 DMA 关闭 1-发送 DMA 打开
0	RDMAE	W/R	接收 DMA 使能 0-接收 DMA 关闭 1-接收 DMA 打开

16.4.22DMA发送数据阈值寄存器（DMATDLR）

- **Name: DMA Transmit Data Level**
- **Size:32bits**
- **Address Offset: 0x50**
- **Read/Write Access: Read/Write**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
预留															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
预留												DMATDLR			

Bit	Name	W/R	Description
31:4	预留	-	预留

<table>
<tr><td>3:0</td><td>DMATDLR</td><td>W/R</td><td>DMA 发送阈值
当发送 FIFO 中数据量等于或小于 DMA 发送阈值，SPI 会对 DMA 发出请求，请求 DMA 向 SPI 发送 FIFO 中写入数据。

设定值参考下表：
<table>
<tr><td>0</td><td>发送 FIFO 中数据数量为 0</td></tr>
<tr><td>1</td><td>发送 FIFO 中数据少于等于 1 个</td></tr>
<tr><td>2</td><td>发送 FIFO 中数据少于等于 2 个</td></tr>
<tr><td>3</td><td>发送 FIFO 中数据少于等于 3 个</td></tr>
<tr><td>4</td><td>发送 FIFO 中数据少于等于 4 个</td></tr>
<tr><td>5</td><td>发送 FIFO 中数据少于等于 5 个</td></tr>
<tr><td>6</td><td>发送 FIFO 中数据少于等于 6 个</td></tr>
<tr><td>7</td><td>发送 FIFO 中数据少于等于 7 个</td></tr>
<tr><td>8</td><td>发送 FIFO 中数据少于等于 8 个</td></tr>
<tr><td>9</td><td>发送 FIFO 中数据少于等于 9 个</td></tr>
<tr><td>10</td><td>发送 FIFO 中数据少于等于 10 个</td></tr>
<tr><td>11</td><td>发送 FIFO 中数据少于等于 11 个</td></tr>
<tr><td>12</td><td>发送 FIFO 中数据少于等于 12 个</td></tr>
<tr><td>13</td><td>发送 FIFO 中数据少于等于 13 个</td></tr>
<tr><td>14</td><td>发送 FIFO 中数据少于等于 14 个</td></tr>
<tr><td>15</td><td>发送 FIFO 中数据少于等于 15 个</td></tr>
</table></td></tr>
</table>

16.4.23DMA接收数据阈值寄存器（DMARDLR）

- **Name: DMA Receive Data Level**
- **Size:32bits**
- **Address Offset: 0x54**
- **Read/Write Access: Read/Write**

<table>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td colspan="16">预留</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="12">预留</td><td colspan="4">DMARDLR</td></tr>
</table>

<table>
<tr><th>Bit</th><th>Name</th><th>W/R</th><th>Description</th></tr>
<tr><td>31:4</td><td>预留</td><td>-</td><td>预留</td></tr>
<tr><td>3:0</td><td>DMARDLR</td><td>W/R</td><td>DMA 接收阈值
当接收 FIFO 中数据量等于或大于 DMA 发送阈值，SPI 会对 DMA 发出请求，请求 DMA 取出 SPI 中接收 FIFO 的数据。

设定值参考下表：
<table>
<tr><td>0</td><td>接收 FIFO 中数据为 1 个及以上</td></tr>
<tr><td>1</td><td>接收 FIFO 中数据为 2 个及以上</td></tr>
<tr><td>2</td><td>接收 FIFO 中数据为 3 个及以上</td></tr>
<tr><td>3</td><td>接收 FIFO 中数据为 4 个及以上</td></tr>
<tr><td>4</td><td>接收 FIFO 中数据为 5 个及以上</td></tr>
<tr><td>5</td><td>接收 FIFO 中数据为 6 个及以上</td></tr>
<tr><td>6</td><td>接收 FIFO 中数据为 7 个及以上</td></tr>
</table></td></tr>
</table>

			7	接收 FIFO 中数据为 8 个及以上
			8	接收 FIFO 中数据为 9 个及以上
			9	接收 FIFO 中数据为 10 个及以上
			10	接收 FIFO 中数据为 11 个及以上
			11	接收 FIFO 中数据为 12 个及以上
			12	接收 FIFO 中数据为 13 个及以上
			13	接收 FIFO 中数据为 14 个及以上
			14	接收 FIFO 中数据为 15 个及以上
			15	接收 FIFO 中数据为 16 个

16.4.24数据寄存器（DR）

- **Name: Data Register**
- **Size:32bits**
- **Address Offset: 0x60**
- **Read/Write Access: read/write**

数据寄存器是1个16bit宽的读写buffer。当对其进行读操作时，数据通过接收FIFO取出。当对其进行写操作时，数据被写入发送FIFO中。只有SSI_EN = 1时才能进行写操作。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
预留															
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
DR															

Bit	Name	W/R	Description
31:16	预留	-	预留
15:0	DR	W/R	SPI 数据寄存器 写数据时数据必须为右对齐数据，读取数据时数据自动右对齐。 Read = 接收数据 FIFO Write = 发送数据 FIFO

16.4.25接收采样延迟寄存器（RX_SAMPLE_DLY）

- **Name: Rx Sample Delay Register**
- **Size:32bits**
- **Address Offset: 0xfc**
- **Read/Write Access: read/write**

该寄存器用于在标准接收采样时间点向后延迟采样点时间。延迟时间以PCLK周期为单位。在SPI使能后可对该寄存器进行读写操作。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
预留								RSD							

Bit	Name	W/R	Description
31:8	预留	-	预留
7:0	RSD	W/R	采样延迟寄存器

			该寄存器用于在标准接收采样时间点向后延迟采样点时间。延迟时间以 PCLK 周期为单位。

17 高速SPI接口

17.1 高速SPI接口简介

本模块为高速SPI接口模块，支持主/从工作模式

17.2 高速SPI接口主要特点

- 高速 SPI 时钟由 HCLK 提供，即 HSPI_CLK = HCLK
- 支持分频
- 支持协议 Motorola Serial Peripheral Interface (SPI)
- 包含硬件实现收发 FIFO，深度为 64
- 独立硬件收发 FIFO，可配收发 FIFO 中断阈值
- 支持主/从，主/从地址不同
- 支持全双工、半双工模式
- 多种收发、错误中断检测
- DMA 支持

17.3 高速SPI接口功能描述

17.3.1 接收/发送FIFO

HSPI master包含2个独立的64x8bit的接收和发送FIFO。
HSPI salve包含2个独立的64x8bit的接收和发送FIFO。

CPU对寄存器DR写操作，数据写入发送FIFO，CPU对寄存器DR读操作，数据由接收FIFO中取出。

接收和发送FIFO有独立的中断阈值设定，当数据符合设定阈值时SPI向CPU发出中断请求。

接收和发送FIFO有独立的DMA阈值设定，当数据符合设定阈值时SPI发出相应DMA请求。

17.3.2 中断类型及中断标志

HSPI master支持以下中断类型：

- Tramsmit Done Interrupt
  发送完成中断，当发送 FIFO 中数据全部发出完成时，产生中断。
- Transmit Interrupt
  发送 FIFO 中断，当发送 FIFO 中的数据量小于设定的下限阈值或大于设定的上限阈值时产生中断。
- Receive Interrupt
  接收 FIFO 中断，当接收 FIFO 中的数据量小于设定的下限阈值或大于设定的上限阈值时产生中断。

HSPI master中断标志：

- Transmit FIFO Arrive Full Interrupt

发送 FIFO 将满标志，当发送 FIFO 中的数据量大于设定的上限阈值时，产生发送 FIFO 将满标志。

- Transmit FIFO Full Interrupt

发送 FIFO 满标志，当发送 FIFO 中的数据量大于设定的上限阈值时，并且发送 FIFO 中的数据量大于 FIFO 深度，产生发送 FIFO 满标志。

- Transmit FIFO Arrive Empty Interrupt

发送 FIFO 将空标志，当发送 FIFO 中数据量小于设定的下限阈值时产生发送 FIFO 将空标志。

- Transmit FIFO Empty Interrupt

发送 FIFO 空标志，当发送 FIFO 中的数据量小于设定的下限阈值时，并且发送 FIFO 中的数据量为 0 时，产生发送 FIFO 空标志。

- Receive FIFO Arrive Full Interrupt

接收 FIFO 将满标志，当接收 FIFO 中的数据量大于设定的上限阈值时，产生接收 FIFO 将满标志。

- Receive FIFO Full Interrupt

接收 FIFO 满标志，当接收 FIFO 中数据量大于设定的上限阈值时，并且接收 FIFO 中的数据量大于 FIFO 深度时，产生接收 FIFO 满标志。

- Receive FIFO Arrive Empty Interrupt

接收 FIFO 将空标志，当接收 FIFO 中的数据量小于设定的下限阈值时，产生接收 FIFO 将空标志。

- Receive FIFO Empty Interrupt

接收 FIFO 空标志，当接收 FIFO 中的数据量小于设定的下限阈值时，并且接收 FIFO 中的数据量为 0，产生接收 FIFO 空标志。

HSPI slave支持以下中断类型：

- Transmit Interrupt

发送 FIFO 中断，当发送 FIFO 中的数据量小于设定的下限阈值或大于设定的上限阈值时产生中断。

- Receive Interrupt

接收 FIFO 中断，当接收 FIFO 中的数据量小于设定的下限阈值或大于设定的上限阈值时产生中断。

HSPI slave中断状态：

- Transmit FIFO Arrive Full Interrupt

发送 FIFO 将满标志，当发送 FIFO 中的数据量大于设定的上限阈值时，产生发送 FIFO 将满标志。

- Transmit FIFO Full Interrupt

发送 FIFO 满标志，当发送 FIFO 中的数据量大于设定的上限阈值时，并且发送 FIFO 中的数据量大于 FIFO 深度，产生发送 FIFO 满标志。

- Transmit FIFO Arrive Empty Interrupt

发送 FIFO 将空标志，当发送 FIFO 中数据量小于设定的下限阈值时产生发送 FIFO 将空标志。

- Transmit FIFO Empty Interrupt

发送 FIFO 空标志，当发送 FIFO 中的数据量小于设定的下限阈值时，并且发送 FIFO 中的数据量为 0 时，产生发送 FIFO 空标志。

- Receive FIFO Arrive Full Interrupt

接收 FIFO 将满标志，当接收 FIFO 中的数据量大于设定的上限阈值时，产生接收 FIFO 将满标志。

- Receive FIFO Full Interrupt

接收 FIFO 满标志，当接收 FIFO 中数据量大于设定的上限阈值时，并且接收 FIFO 中的数据量大于 FIFO 深度时，产生接收 FIFO 满标志。

- Receive FIFO Arrive Empty Interrupt

接收 FIFO 将空标志，当接收 FIFO 中的数据量小于设定的下限阈值时，产生接收 FIFO 将空标志。

- Receive FIFO Empty Interrupt

接收 FIFO 空标志，当接收 FIFO 中的数据量小于设定的下限阈值时，并且接收 FIFO 中的数据量为 0，产生接收 FIFO 空标志。

17.4 寄存器描述

17.4.1 地址映射表

偏移地址	寄存器名称	宽度（bit）	复位值
0x00	SLV _CON	32	0x00000000
0x04	SLV_STAT	32	0x00000606
0x08	SLV_TX	32	0x00000000
0x0C	SLV_RX	32	0x00000000
0x10	SLV_THCON	32	0x003f003f
0x14	SLV_TINT	32	0x00000000
0x18	SLV_FIFO_STATUS	32	0x00000000
0x1C	SLV_RINT	32	0x00000000
0x20	MASTER_CTRL0	32	0x01000000
0x24	预留	32	0x00000000
0x28	MASTER_FLOW_STATUS	32	0x00000000
0x2C	MASTER_FIFO_CTRL	32	0x003f003f
0x30	MASTER_RDATA	32	0x00000000
0x34	MASTER_TDATA	32	0x00000000
0x38	MASTER_STATUS	32	0x00000606
0x3C	MASTER_CTRL1	32	0x00000000
0x40	MASTER_FIFO_STATUS	32	0x00000000
0x44	MASTER_DMA_CTRL	32	0x00000000
0x48	MASTER_TINT	32	0x00000000
0x4C	MASTER_RINT	32	0x00000000

17.4.2 从配置寄存器（SLAVE_CON）

31	30	29	28	27	26	25	24
预留							
ro							

23	22	21	20	19	18	17	16
预留	tx_int_en	rx_int_en	sample_sel		tx_fifo_thr		
ro	rw	rw	rw		rw		

15	14	13	12	11	10	9	8
tx_fifo_thr				tx_dma_en	rx_fifo_thr		
ro				rw	rw		

7	6	5	4	3	2	1	0
rx_fifo_thr				rx_dma_en	int_en	rx_start	tx_start
rw				rw	rw	rw	rw

Bit	Name	W/R	Description
31-23	预留	--	
22	tx_int_en	RW	tx中断使能

21	rx_int_en	RW	rx中断使能
20-19	sample_sel	RW	{cpol,cpha}=2'b00,2'b01,b'b10,2'b11
18-12	tx_fifo_thr	RW	dma启动时配置的tx fifo阈值。
11	tx_dma_en	RW	tx_dma 使能开关
10-4	rx_fifo_thr	RW	dma启动时配置的rx fifo阈值。
3	rx_dma_en	RW	rx_dma 使能开关
2	int_en	RW	中断使能
1	rx_start	RW	启动接收功能，rx_fifo开始接收数据
0	tx_start	RW	启动发送功能，tx_fifo开始发送数据

17.4.3 从状态寄存器（SLV_STAT）

31	30	29	28	27	26	25	24
预留							
ro							
23	22	21	20	19	18	17	16
预留							
ro							
15	14	13	12	11	10	9	8
预留		push_af_rx	push_full_rx	push_error_rx	pop_empty_rx	pop_ae_rx	pop_error_rx
ro		ro	ro	ro	ro	ro	ro
7	6	5	4	3	2	1	0
预留		push_af_tx	push_full_tx	push_error_tx	pop_empty_tx	pop_ae_tx	pop_error_tx
ro		ro	ro	ro	ro	ro	ro

Bit	Name	W/R	Description
31-14	预留	--	
13	push_af_rx	RO	rx fifo push达到阈值
12	push_full_rx	RO	rx fifo push满
11	push_error_rx	RO	rx fifo push错误
10	pop_empty_rx	RO	rx fifo空
9	pop_ae_rx	RO	rx fifo接收达到阈值
8	pop_error_rx	RO	rx fifo错误
7-6	预留	--	
5	push_af_tx	RO	tx fifo push达到阈值
4	push_full_tx	RO	tx fifo push满
3	push_error_tx	RO	tx fifo push错误
2	pop_empty_tx	RO	tx fifo空
1	pop_ae_tx	RO	tx fifo发送达到阈值
0	pop_error_tx	RO	tx fifo错误

17.4.4 从发送FIFO写寄存器（SLV_TX）

31	30	29	28	27	26	25	24
wdata							
wo							
23	22	21	20	19	18	17	16
wdata							
wo							

15	14	13	12	11	10	9	8
wdata							
wo							
7	6	5	4	3	2	1	0
wdata							
wo							

Bit	Name	W/R	Description
31-0	wdata	WO	向tx_fifo写入将要发送的数据

17.4.5 从接收FIFO读寄存器（SLV_RX）

31	30	29	28	27	26	25	24
rdata							
ro							
23	22	21	20	19	18	17	16
rdata							
ro							
15	14	13	12	11	10	9	8
rdata							
ro							
7	6	5	4	3	2	1	0
rdata							
ro							

Bit	Name	W/R	Description
31-0	rdata	RO	读rx_fifo数据

17.4.6 从FIFO配置寄存器（SLV_THCON）

31	30	29	28	27	26	25	24
预留		rx_ae_th					
ro		rw					
23	22	21	20	19	18	17	16
预留		rx_af_th					
ro		rw					
15	14	13	12	11	10	9	8
预留		tx_ae_th					
ro		rw					
7	6	5	4	3	2	1	0
tx_fifo_clr	rx_fifo_clr	tx_af_th					
rw	rw	rw					

Bit	Name	W/R	Description
31-30	预留	--	
29-24	rx_ae_th	RW	rx fifo 将空门限值
23-22	预留	--	
21-16	rx_af_th	RW	rx fifo 将满门限值
15-14	预留	--	
13-8	tx_ae_th	RW	tx fifo 将空门限值
7	tx_fifo_clr	RW	写1清空tx fifo
6	rx_fifo_clr	RW	写1清空rx fifo
5-0	tx_af_th	RW	tx fifo 将满门限值

17.4.7 从发送中断寄存器（SLV_TINT）

31	30	29	28	27	26	25	24
预留							
ro							

23	22	21	20	19	18	17	16
预留							
ro							

15	14	13	12	11	10	9	8
预留							
ro							

7	6	5	4	3	2	1	0
预留				tx_full_int	tx_af_int	tx_empty_int	tx_ae_int
ro				rc	rc	rc	rc

Bit	Name	W/R	Description
31-4	预留	--	
3	tx_full_int	RC	发送fifo满中断
2	tx_af_int	RC	发送fifo将满中断
1	tx_empty_int	RC	发送fifo空中断
0	tx_ae_int	RC	发送fifo将空中断

17.4.8 从FIFO状态寄存器（SLV_FIFO_STATUS）

31	30	29	28	27	26	25	24
预留							
ro							

23	22	21	20	19	18	17	16
预留							
ro							

15	14	13	12	11	10	9	8
预留		rx_pop_depth					
ro		ro					

7	6	5	4	3	2	1	0
rx_pop_depth	tx_push_depth						
ro	ro						

Bit	Name	W/R	Description
31-14	预留	--	
13-7	rx_pop_depth	RO	rx fifo中数据个数
6-0	tx_push_depth	RO	tx fifo中数据个数

17.4.9 从接收中断寄存器（SLV_RINT）

31	30	29	28	27	26	25	24
预留							
ro							

23	22	21	20	19	18	17	16
预留							

15	14	13	12	11	10	9	8
预留							
ro							

7	6	5	4	3	2	1	0
预留				rx_full_int	rx_af_int	rx_empty_int	rx_ae_int
ro				rc	rc	rc	rc

Bit	Name	W/R	Description
31-4	预留	--	
3	rx_full_int	RC	接收fifo满中断
2	rx_af_int	RC	接收fifo将满中断
1	rx_empty_int	RC	接收fifo空中断
0	rx_ae_int	RC	接收fifo将空中断

17.4.10 主控制寄存器0（MASTER_CTRL0）

31	30	29	28	27	26	25	24
预留						default_output	master_enable
ro						rw	rw

23	22	21	20	19	18	17	16
预留			rx_dma_en	预留			tx_dma_en
ro			rw	ro			rw

15	14	13	12	11	10	9	8
rx_done_int_en	tx_done_int_en	int_en	modsel			lsbfe	cpol
rw	rw	rw	rw			rw	rw

7	6	5	4	3	2	1	0
cpha	预留				mdiv_en	tx_enable	master_busy
rw	ro				rw	rw	rc

Bit	Name	W/R	Description
31-26	预留	--	
25	default_output	RW	发送第一个word时的default值
24	master_enable	RW	master使能。
23:21	预留	--	
20	rx_dma_en	RW	接收dma使能开关
19:17	预留	--	
16	tx_dma_en	RW	发送dma使能开关
15	rx_done_int_en	RW	接收中断使能 1：当接收fifo深度大于rx_af_th或小于rx_ae_th时，会产生spi中断 0：不产生中断
14	tx_done_int_en	RW	发送中断使能 1：当发送fifo深度大于tx_af_th或小于tx_ae_th时，会产生spi中断 0：不产生中断
13	int_en	RW	spi中断使能信号 0：不产生中断

			1：当spi_master结束或者tx/rx 出现错误时，产生中断信号
12:10	modsel	RW	模式选择： 3'b000/001:单线全双工模式。
9	lsbfe	RW	大小端选择 0：小端，左高右低。 1：大端
8	cpol	RW	极性选择
7	cpha	RW	相位选择
6:3	预留	RO	
2	mdiv_en	RW	分频使能信号。不使能情况下，默认2分频。
1	tx_enable	RW	tx使能开关，打开之后，spi_master可以开启TX功能。
0	master_busy	RC	当前配置了MASTER_CTRL0寄存器标志位，读清寄存器。

17.4.11 主状态寄存器（MASTER_FLOW_STATUS）

31	30	29	28	27	26	25	24
预留							
ro							

23	22	21	20	19	18	17	16
预留							
ro							

15	14	13	12	11	10	9	8
预留							
ro							

7	6	5	4	3	2	1	0
预留							spi_done
ro							rc

Bit	Name	W/R	Description
31-1	预留	--	
0	spi_done	RC	SPI传输结束

17.4.12 主FIFO控制寄存器（MASTER_FIFO_CTRL）

31	30	29	28	27	26	25	24
预留		rx_ae_th					
ro		rw					

23	22	21	20	19	18	17	16
预留		rx_af_th					
ro		rw					

15	14	13	12	11	10	9	8
预留		tx_ae_th					
ro		rw					

7	6	5	4	3	2	1	0
tx_fifo_clr	rx_fifo_clr	tx_af_th					
rw	rw	rw					

Bit	Name	W/R	Description

31-30	预留	--	
29-24	rx_ae_th	RW	接收FIFO将空门限值
23-22	预留	--	
21-16	rx_af_th	RW	接收FIFO将满门限值
15-14	预留	--	
13-8	tx_ae_th	RW	发送FIFO将空门限值
7	tx_fifo_clr	RW	写1清空发送FIFO
6	rx_fifo_clr	RW	写1清空接收FIFO
5-0	tx_af_th	RW	发送FIFO将满门限值

17.4.13 主接收FIFO读寄存器（MASTER_RDATA）

31	30	29	28	27	26	25	24
rdata							
ro							
23	22	21	20	19	18	17	16
rdata							
ro							
15	14	13	12	11	10	9	8
rdata							
ro							
7	6	5	4	3	2	1	0
rdata							
ro							

Bit	Name	W/R	Description
31-0	rdata	RO	SPI Master读FIFO数据入口

17.4.14 主发送FIFO读寄存器（MASTER_TDATA）

31	30	29	28	27	26	25	24
tdata							
wo							
23	22	21	20	19	18	17	16
tdata							
wo							
15	14	13	12	11	10	9	8
tdata							
wo							
7	6	5	4	3	2	1	0
tdata							
wo							

Bit	Name	W/R	Description
31-0	tdata	WO	SPI Master写FIFO数据入口

17.4.15 主状态寄存器（MASTER_STATUS）

31	30	29	28	27	26	25	24
预留							
ro							
23	22	21	20	19	18	17	16

预留							
ro							

15	14	13	12	11	10	9	8
预留		push_af_rx	push_full_rx	push_error_rx	pop_empty_rx	pop_ae_rx	pop_error_rx
ro		ro	ro	ro	ro	ro	ro

7	6	5	4	3	2	1	0
预留		push_af_tx	push_full_tx	push_error_tx	pop_empty_tx	pop_ae_tx	pop_error_tx
ro		ro	ro	ro	ro	ro	ro

Bit	Name	W/R	Description
31-14	预留	--	
13	push_af_rx	RO	接收fifo push达到阈值
12	push_full_rx	RO	接收fifo push满
11	push_error_rx	RO	接收fifo push错误
10	pop_empty_rx	RO	接收fifo发送空
9	pop_ae_rx	RO	接收fifo发送达到阈值
8	pop_error_rx	RO	接收fifo错误
7-6	预留	--	
5	push_af_tx	RO	发送fifo push达到阈值
4	push_full_tx	RO	发送fifo push满
3	push_error_tx	RO	发送fifo push错误
2	pop_empty_tx	RO	发送fifo发送空
1	pop_ae_tx	RO	发送fifo发送达到阈值
0	pop_error_tx	RO	发送fifo错误

17.4.16 主控制寄存器1（MASTER_CTRL1）

31	30	29	28	27	26	25	24
预留							
ro							

23	22	21	20	19	18	17	16
预留				freq_div			
ro				rw			

15	14	13	12	11	10	9	8
freq_div						rx_dlen	
rw						rw	

7	6	5	4	3	2	1	0
rx_dlen							
rw							

Bit	Name	W/R	Description
31-20	预留	--	
19-10	freq_div	RW	时钟分频： 0,1：2分频 2： 4分频 3： 6分频 4； 8分频 …… 512 :1024分频 ……

			分频数 = freq_div * 2。
9-0	rx_dlen	RW	master接收数据长度， 0: 1 byte 1: 2 byte 2: 3 byte 3：4 byte 4：5 byte …… >=127: 128 byte

17.4.17 主FIFO状态寄存器（MASTER_FIFO_STATUS）

31	30	29	28	27	26	25	24
预留							
ro							

23	22	21	20	19	18	17	16
预留							
ro							

15	14	13	12	11	10	9	8
预留		rx_pop_depth					
ro		ro					

7	6	5	4	3	2	1	0
rx_pop_depth	tx_push_depth						
ro	ro						

Bit	Name	W/R	Description
31-14	预留	--	
13-7	rx_pop_depth	RO	接收FIFO中数据量
6-0	tx_push_depth	RO	发送FIFO中数据量

17.4.18 主DMA控制寄存器（MASTER_DMA_CTRL）

31	30	29	28	27	26	25	24
预留							
ro							

23	22	21	20	19	18	17	16
预留							
ro							

15	14	13	12	11	10	9	8
预留		rx_fifo_thr					
ro		rw					

7	6	5	4	3	2	1	0
rx_fifo_thr	tx_fifo_thr						
rw	rw						

Bit	Name	W/R	Description
31:14	预留	--	
13:7	rx_fifo_thr	RW	DMA访问接收FIFO门限值
6:0	tx_fifo_thr	RW	DMA访问发送FIFO门限值

17.4.19 主发送中断寄存器（MASTER_TINT）

31	30	29	28	27	26	25	24
预留							
ro							

23	22	21	20	19	18	17	16
预留							
ro							

15	14	13	12	11	10	9	8
预留							
ro							

7	6	5	4	3	2	1	0
预留				tx_full_int	tx_af_int	tx_empty_int	tx_ae_int
ro				rc	rc	rc	rc

Bit	Name	W/R	Description
31:4	预留	--	
3	tx_full_int	RC	发送FIFO满中断
2	tx_af_int	RC	发送FIFO将满中断
1	tx_empty_int	RC	发送FIFO空中断
0	tx_ae_int	RC	发送FIFO将空中断

17.4.20 主接收中断寄存器（MASTER_RINT）

31	30	29	28	27	26	25	24
预留							
ro							

23	22	21	20	19	18	17	16
预留							
ro							

15	14	13	12	11	10	9	8
预留							
ro							

7	6	5	4	3	2	1	0
预留				rx_full_int	rx_af_int	rx_empty_int	rx_ae_int
ro				rc	rc	rc	rc

Bit	Name	W/R	Description
31:4	预留	--	
3	rx_full_int	RC	接收FIFO满中断
2	rx_af_int	RC	接收FIFO将满中断
1	rx_empty_int	RC	接收FIFO空中断
0	rx_ae_int	RC	接收FIFO将空中断

18 DMA控制器（DMAC）

18.1 DMA简介

直接存储器存取(DMA)用来提供在外设和存储器之间或者存储器和存储器之间的高速数据传输。无须CPU干预，数据可以通过DMA快速地移动，这就节省了CPU的资源来做其他操作。

DMA控制器有4个通道，专门用来管理内存到内存、内存到外设、外设到内存的访问请求。

18.2 DMA主要特性

- 4 个独立的可配置通道
- 支持多种宽度数据传输
- 每个通道有各自独立中断信号及标记
- 支持内存到内存、内存到外设、外设到内存之间的传输
- 支持 Multi-Block 传输

18.3 外设类型

使用DMA传输的外设可分为两类外设：

- 存储器外设
- 非存储器外设

存储器外设：

存储器外设与DMA控制器间不需要"DMA握手接口"，因此当DMA通道使能后，DMA不需要请求信号即可开始DMA传输。同时由于无握手接口，所以SRC_MSIZE和DEST_MSIZE对存储器外设无限制作用。

非存储器外设：

非存储器外设，如：UART、I2C、SPI等均包含与DMA直接进行通信的请求信号，即DMA握手接口。非存储器外设进行DMA传输时，会通过与DMA直接的握手接口对数据的传输进行控制。DMA接口分为两种：硬握手接口和软握手接口

18.4 DMA握手接口

DMA提供两种DMA请求握手接口：

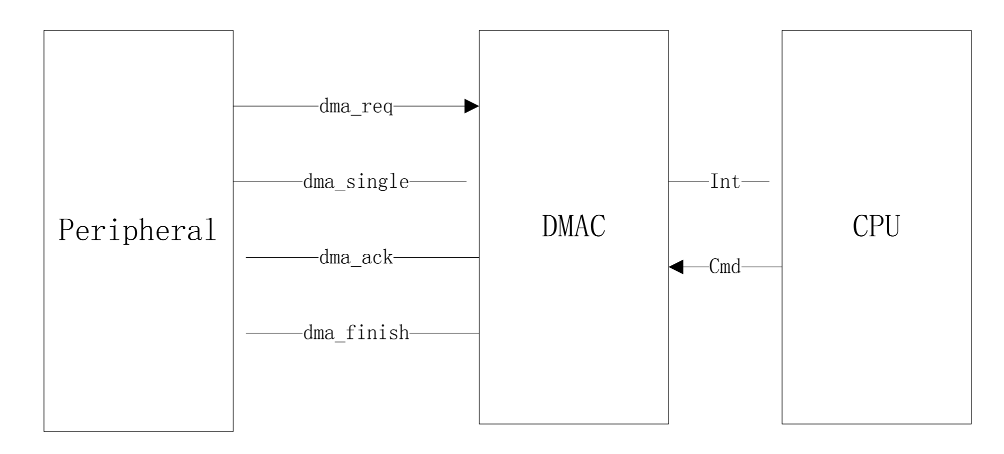
- 硬握手接口
- 软握手接口

用户可以通过DMA寄存器配置各通道寄存器选择握手接口方式。

18.4.1 DMA硬握手接口

当用户配置对应外设DMA使能、配置相应触发条件及对应DMA通道为硬握手信号后，DMA请求信号触发将由硬件根据条件自动完成。

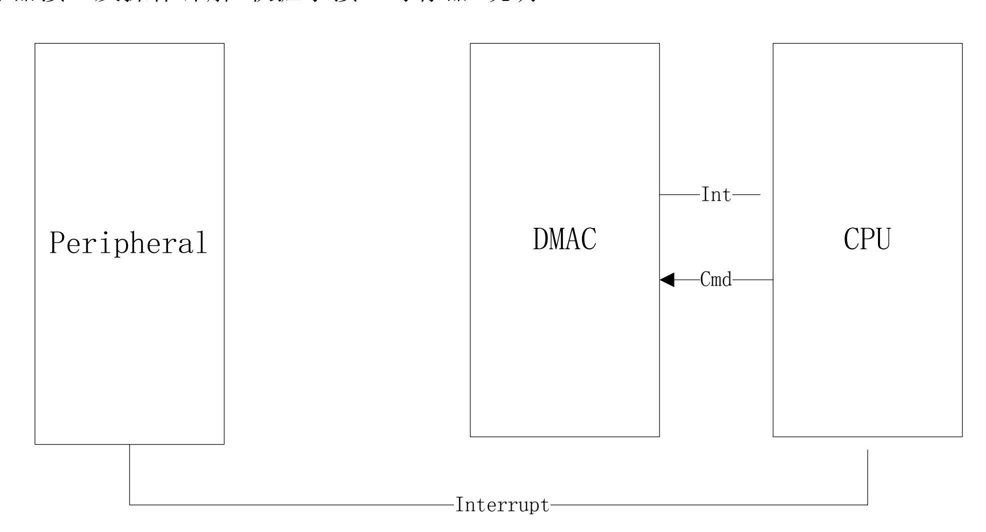
DMA控制器硬握手接口通过dma_req、dma_single、dma_ack和dma_ finish五个信号同相应外设进行交互。SCPU中具有DMA功能的外设均包含以下接口信号。。



Name	Description
dma_req	Burst transaction request from peripheral 外设向 DMA 发出突发传输请求
dma_single	Single transfer status 外设单次传输状态
dma_ack	DMA 应答信号 DMA 完成 1 次在 AHB 总线上传输（突发（Burst）或单次（single））后对外设响应信号
dma_finish	DMA 传输完成 DMA 完成块（BLOCK）传输后对外设响应信号

18.4.2 DMA软握手接口

当外设没有DMA接口或用户不使用外设接口时，可以使用DMA软握手接口操作DMA数据传输。寄存器接口及操作详解“软握手接口寄存器”说明。



- DMA 请求寄存器：ReqSrcReg/ReqDstReg
  产生 DMA 请求，如 ReqSrcReg[x]置 1，即源设备对 DMA 通道 x 产生 1 次请求
- DMA 单次请求寄存器：SglReqSrcReg/SglReqDstReg
  设定 DMA 请求类型，如 ReqSrcReg[x]为 0，则源设备对 DMA 通道 x 产生“突发”传输请求，为 1，则源设备对 DMA 通道 x 产生“单次”传输请求
- DMA 最后请求寄存器：LstSrcReg/LstDstReg
  设定本次块传输的最后一次 DMA 请求，如 LstSrcReg[x]为 1，表示此次 DMA 请求为源设备对 DMA 通道 x 的最后一次请求
- 寄存器写入顺序：

在ReqSrcReg/ReqDstReg写入前需要保证SglReqSrcReg/SglReqDstReg和LstSrcReg/LstDstReg已经正确写入

寄存器使用示例：

- 源设备对 DMA 通道 2 产生 1 次“突发”数据传输：
  SglReqSrcReg[2] = 0;
  LstSrcReg[2] = 0;
  ReqSrcReg[2] = 1;
- 源设备对 DMA 通道 2 产生 1 次“单次”数据传输：
  SglReqSrcReg[2] = 1;
  LstSrcReg[2] = 0;
  ReqSrcReg[2] = 1;
- 源设备对 DMA 通道 2 产生在此次 DMA 块传输的最后 1 次“单次”数据传输
  SglReqSrcReg[2] = 1;
  LstSrcReg[2] = 1;
  ReqSrcReg[2] = 1;

18.5 DMA中断

18.5.1 中断类型

DMA每个通道均包含5种中断类型：

- 块传输完成中断（IntBlock – Block Transfer Complete Interrupt）
  DMA 通道对目标设备的块传输数据完成后，中断置位。
- 目标数据传输完成中断（IntDstTran – Destination Transaction Complete Interrupt）
  DMA 通道在 AHB 总线上完成 1 次（“单次”或“突发”）对目标设备的数据传输后，中断置位。
- 错误中断（IntErr – Error Interrupt）
  在 DMA 传输过程中，DMA 接收到来自 AHB 总线上的错误应答后，中断置位。
- 源数据传输完成中断（IntSrcTran – Source Transaction Complete Interrupt）
  DMA 通道在 AHB 总线上完成 1 次（“单次”或“突发”）对源设备的数据传输后，中断置位。
- DMA 传输完成中断（IntTfr – DMA Transfer Complete Interrupt）
  DMA 完成对目标设备数据传输后，中断置位。

18.5.2 中断寄存器

5组中断寄存器对应每种中断类型，其中每个寄存器中的bit位对应1个通道。

- 原（Raw）中断状态寄存器：RawBlock, RawDstTran, RawErr, RawSrcTran, RawTfr
  DMA 中断事件被保存在原中断状态寄存器中

- 中断状态寄存器：StatusBlock, StatusDstTran, StatusErr, StatusSrcTran, StatusTfr
  未被中断屏蔽寄存器屏蔽的中断状态保存到中断状态寄存器中

- 中断屏蔽寄存器：MaskBlock, MaskDstTran, MaskErr, MaskSrcTran, MaskTfr
  打开或关闭 DMA 通道中断状态

- 中断清除寄存器：ClearBlock, ClearDstTran, ClearErr, ClearSrcTran, ClearTfr
  对中断控制寄存器写“1”清除对应通道产生中断

- StatusInt

个中断标志位为中断状态寄存器组中断标记为“或”运算后结果。

18.6 数据传输规则

DMA寄存器与数据类型及传输量有关的3个参数分别是：

传输数据块大小：BLOCK_TS
突发数据传输大小：SRC_MSIZE、DEST_MSIZE
数据位宽：SRC_TR_WIDTH、DST_TR_WIDTH

18.6.1 存储器到存储器

存储器到存储器的DMA传输由DMA BLOCK_TS和SRC_TR_WIDTH来决定传输量，DMA与存储器外设无握手接口，数据传输过程中不会对数据传输进行控制。

总传输量Bytes = BLOCK_TS * (SRC_TR_WIDTH / 8)

18.6.2 存储器到外设/外设到存储器

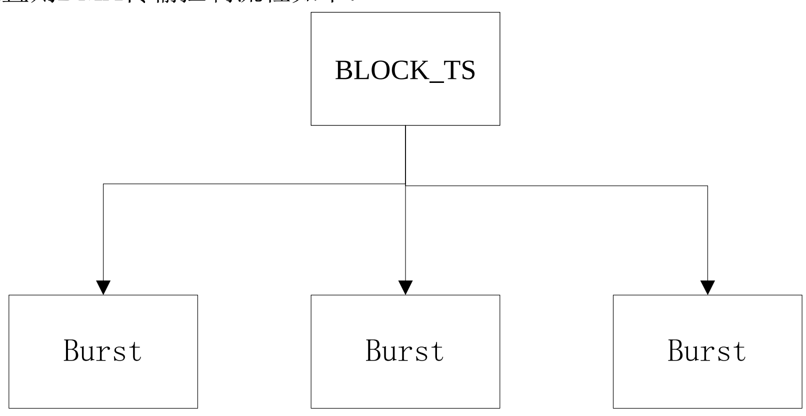
在存储器到外设/外设到存储器情况下，存储器与DMA直接无握手接口库，存储器到DMA与存储器到存储器间情况一样。

DMA与非存储器外设的数据传输会受到FIFO大小的影响，因此DMA会对数据传输进行控制。DMA使用SRC_MSIZE/DEST_MSIZE来限制1次“突发”（Burst）传输的数据量大小。

- 若 DMA BLOCK_TS 是 SRC_MSIZE/DEST_MSIZE 的整数倍，则整个传输过程中仅产生“突发”传输。

  示例参数配置：
  DMA BLOCK_TS = 12
  SRC_MSIZE/DEST_MSIZE = 4

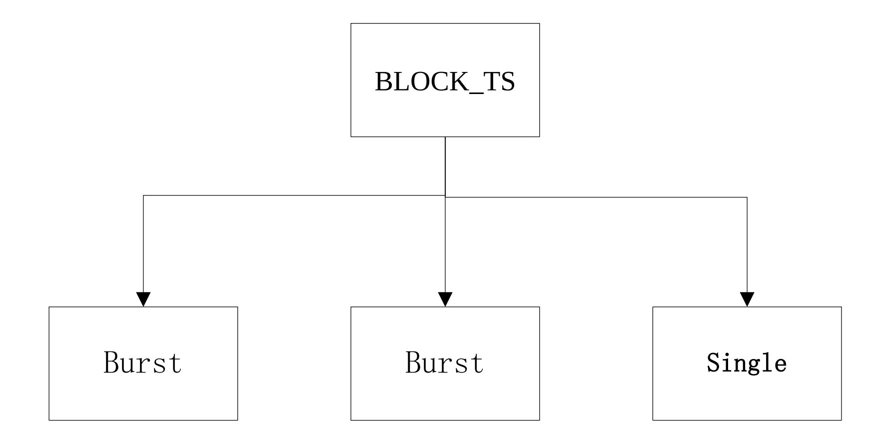
  根据以上配置则DMA传输控制流程如下：



- 若 DMA BLOCK_TS 不是 SRC_MSIZE/DEST_MSIZE 的整数倍，则在最后剩余的传输中使用“单次”（Single）传输完成。“单次”传输每次仅传输 SRC_TR_WIDTH/DST_TR_WIDTH 单个数据

  示例参数配置：
  DMA BLOCK_TS = 12
  SRC_MSIZE/DEST_MSIZE = 5

  根据以上配置则DMA传输控制流程如下：

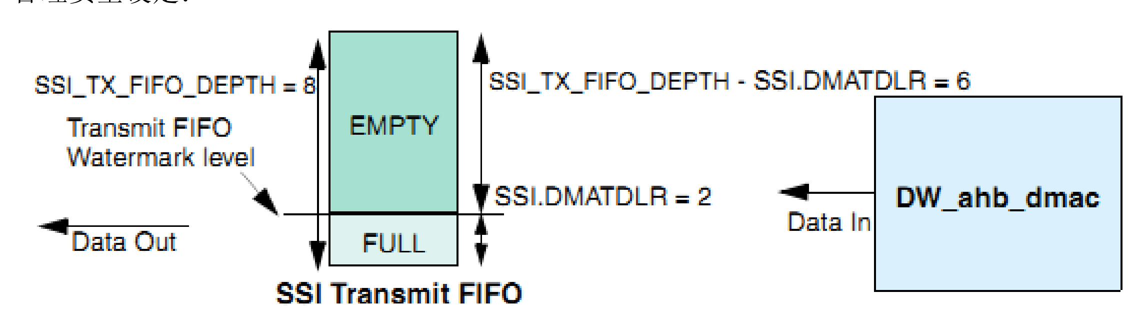


18.6.3 SRC_MSIZE/DEST_MSIZE参数置设定

在非存储器外设中会使用到 SRC_MSIZE/DEST_MSIZE 参数。
SRC_MSIZE/DEST_MSIZE 与非存储器外设接收和发送 FIFO 设定相关。

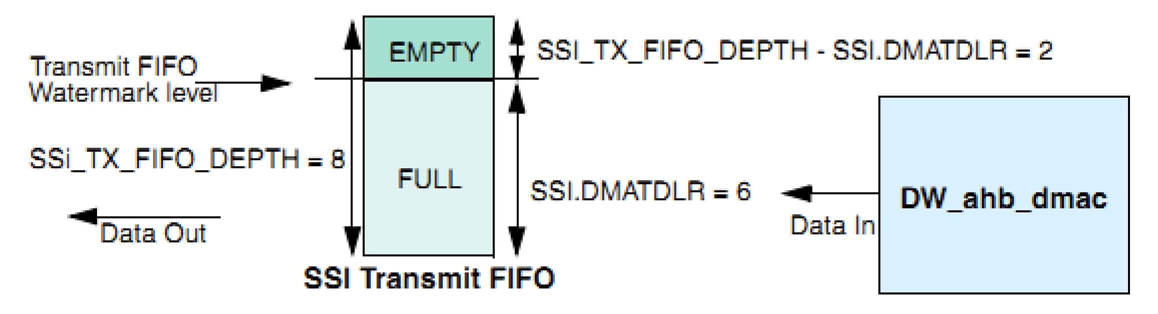
- 当非存储器外设作为目标设备时，非存储器外设发送 FIFO 中数据达到设定阈值后，向 DMA 发出请求，此时 DMA 会根据 DEST _MSIZE 中的数据量向发送 FIFO 中写数据。此时发送 FIFO 剩余的空间应大于 DEST_MSIZE 值，以满足 DMA 通道 1 次搬运的数据量，如果小于则会引起对应外设 FIFO 上溢错误。

合理安全设定：



参数	注释
SSI.DMATDLR = 2	SSI 外设发送 FIFO 阈值
DMA.CTLx.DEST_MSIZE = 6 SSI_TX_FIFO_DEPTH -SSI.DMATDLR = 6	DEST_MSIZE 等于 FIFO 剩余空间大小
SSI _TX_FIFO_DEPTH = 8	发送 FIFO 总深度
DMA.CTLx.BLOCK_TS = 30	块大小

不合理设定：
DMA.CTLx.DEST_MSIZE > SSI_TX_FIFO_DEPTH -SSI.DMATDLR



参数	注释

SSI.DMATDLR = 6	SSI 外设发送 FIFO 阈值
DMA.CTLx.DEST_MSIZE = 4 SSI_TX_FIFO_DEPTH -SSI.DMATDLR = 2	DEST_MSIZE 小于 FIFO 剩余空间大小
SSI _TX_FIFO_DEPTH = 8	发送 FIFO 总深度
DMA.CTLx.BLOCK_TS = 30	块大小

- 当非存储器外设作为源设备时，非存储器外设接收 FIFO 中数据达到设定阈值后，向 DMA 发出请求，此时 DMA 会根据 SRC_MSIZE 中的数据量由接收 FIFO 中取数据。因此接收 FIFO 中设定的阈值应大于 SRC_MSIZ 值，以满足 DMA 通道 1 次读取数据量，如果小于 1 次读取数据量会导致相应外设产生 FIFO 下溢中断。

18.7 Multi-Block模式

表 18-1 DMA Multi-Block模式

传输类型	LLPx. LOC	LLP_SRC_EN	RELOAD _SRC	LLP_DST_EN	RELOAD_DST	CTLx, LLPx Update	SARx Update	DARx Update	Write Back
1.Single-block or last transfer of multi-block	0	0	0	0	0	手动编程	None	None	No
2.Auto-reload multi-block transfer with contiguous SAR	0	0	0	0	1	自动加载初始值	Contiguous	Auto-Reload	No
3.Auto-reload multi-block transfer with contiguous DAR	0	0	1	0	0	自动加载初始值	Auto-Reload	Contiguous	No
4.Auto-reload multi-block transfer	0	0	1	0	1	自动加载初始值	Auto-Reload	Auto-Reload	No
5. Single-block or last transfer of multi-block	1	0	0	0	0	手动编程	None	None	Yes
6. Linked list multi-block transfer with contiguous SAR	1	0	0	1	0	自动加载下一链表项	Contiguous	Linked List	Yes
7. Linked list multi-block transfer with auto-reload SAR	1	0	1	1	0	自动加载下一链表项	Auto-Reload	Linked List	Yes
8. Linked list multi-block transfer with contiguous DAR	1	1	0	0	0	自动加载下一链表项	Linked List	Contiguous	Yes
9. Linked list multi-block transfer with auto-reload	1	1	0	0	1	自动加载下一链表项	Linked List	Auto-Reload	Yes

DAR									
10. Linked list multi-block transfer	1	1	0	1	0	自动加载下一链表项	Linked List	Linked List	Yes

18.8 DMA寄存器描述

18.8.1 地址映射表

DMA基地址列表

地址范围	基地址	外设	总线
0x4000_0800-0x4000_0BFF	0x4000_0800	DMA	AHB

表 18-2 DMA寄存器表

偏移地址	寄存器名称	宽度（bit）	复位值	寄存器域
0x000	SAR0	32	0x00000000	DMA_Channel_0
0x004	预留	32	0x00000000	
0x008	DAR0	32	0x00000000	
0x00C	预留	32	0x00000000	
0x010	LLP0	32	0x00000000	
0x014	预留	32	0x00000000	
0x018	CTL0_L	32	0x00304801	
0x01C	CTL0_H	32	0x00000002	
0x020 - 0x03F	预留 x 8	32 x 8	0x00000000	
0x040	CFG0	32	0x00000E00	
0x044 - 0x057	预留 x 5	32 x 5	0x00000000	
0x058	SAR1	32	0x00000000	DMA_Channel_1
0x05C	预留	32	0x00000000	
0x060	DAR1	32	0x00000000	
0x064	预留	32	0x00000000	
0x068	LLP1	32	0x00000000	
0x06C	预留	32	0x00000000	
0x070	CTL1_L	32	0x00304801	
0x074	CTL1_H	32	0x00000002	
0x078 - 0x097	预留 x 8	32 x 8	0x00000000	
0x098	CFG1	32	0x00000E20	
0x09C - 0x0AF	预留 x 5	32 x 5	0x00000000	
0x0B0	SAR2	32	0x00000000	DMA_Channel_2
0x0B4	预留	32	0x00000000	
0x0B8	DAR2	32	0x00000000	
0x0BC	预留	32	0x00000000	
0x0C0	LLP2	32	0x00000000	
0x0C4	预留	32	0x00000000	
0x0C8	CTL2_L	32	0x00304801	
0x0CC	CTL2_H	32	0x00000002	
0x0D0 - 0x0EF	预留 x 8	32 x 8	0x00000000	
0x0F0	CFG2	32	0x00000E40	
0x0F4 - 0x107	预留 x 5	32 x 5	0x00000000	
0x108	SAR3	32	0x00000000	DMA_Channel_3
0x10C	预留	32	0x00000000	
0x110	DAR3	32	0x00000000	
0x114	预留	32	0x00000000	

0x118	LLP3	32	0x00000000	
0x11C	预留	32	0x00000000	
0x120	CTL3_L	32	0x00304801	
0x124	CTL3_H	32	0x00000002	
0x128 - 0x147	预留 x 8	32 x 8	0x00000000	
0x148	CFG3	32	0x00000E60	
0x14C - 0x15F	预留 x 5	32 x 5	0x00000000	
0x160 - 0x2BF	预留 x 88	32 x 88	0x00000000	预留
0x2C0	RawTfr	32	0x00000000	DMA_Interrupt
0x2C4	预留	32	0x00000000	
0x2C8	RawBlock	32	0x00000000	
0x2CC	预留	32	0x00000000	
0x2D0	RawSrcTran	32	0x00000000	
0x2D4	预留	32	0x00000000	
0x2D8	RawDstTran	32	0x00000000	
0x2DC	预留	32	0x00000000	
0x2E0	RawErr	32	0x00000000	
0x2E4	预留	32	0x00000000	
0x2E8	StatusTfr	32	0x00000000	
0x2EC	预留	32	0x00000000	
0x2F0	StatusBlock	32	0x00000000	
0x2F4	预留	32	0x00000000	
0x2F8	StatusSrcTran	32	0x00000000	
0x2FC	预留	32	0x00000000	
0x300	StatusDstTran	32	0x00000000	
0x304	预留	32	0x00000000	
0x308	StatusErr	32	0x00000000	
0x30C	预留	32	0x00000000	
0x310	MaskTfr	32	0x00000000	
0x314	预留	32	0x00000000	
0x318	MaskBlock	32	0x00000000	
0x31C	预留	32	0x00000000	
0x320	MaskSrcTran	32	0x00000000	
0x324	预留	32	0x00000000	
0x328	MaskDstTran	32	0x00000000	
0x32C	预留	32	0x00000000	
0x330	MaskErr	32	0x00000000	
0x334	预留	32	0x00000000	
0x338	ClearTfr	32	0x00000000	
0x33C	预留	32	0x00000000	
0x340	ClearBlock	32	0x00000000	
0x344	预留	32	0x00000000	
0x348	ClearSrcTran	32	0x00000000	
0x34C	预留	32	0x00000000	
0x350	ClearDstTran	32	0x00000000	
0x354	预留	32	0x00000000	
0x358	ClearErr	32	0x00000000	
0x35C	预留	32	0x00000000	
0x360	StatusInt	32	0x00000000	
0x364	预留	32	0x00000000	
0x368	ReqSrcReg	32	0x00000000	Soft_HandShake
0x36C	预留	32	0x00000000	
0x370	ReqDstReg	32	0x00000000	

0x374	预留	32	0x00000000	
0x378	SglReqSrcReg	32	0x00000000	
0x37C	预留	32	0x00000000	
0x380	SglReqDstReg	32	0x00000000	
0x384	预留	32	0x00000000	
0x388	LstSrcReg	32	0x00000000	
0x38C	预留	32	0x00000000	
0x390	LstDstReg	32	0x00000000	
0x394	预留	32	0x00000000	
0x398	DmaCfgReg	32	0x00000000	DMA_Control
0x39C	预留	32	0x00000000	
0x3A0	ChEnReg	32	0x00000000	
0x3A4	预留	32	0x00000000	

DMA寄存器区域划分表

寄存器域地址范围	寄存器域基地址	寄存器域	外设
0x4000_0800-0x4000_0857	0x4000_0800	DMA_Channel_0	DMA
0x4000_0858-0x4000_08AF	0x4000_0858	DMA_Channel_1	
0x4000_08B0-0x4000_0907	0x4000_08B0	DMA_Channel_2	
0x4000_0908-0x4000_095F	0x4000_0908	DMA_Channel_3	
0x4000_0960-0x4000_0ABF	0x4000_0960	预留	
0x4000_0AC0-0x4000_0B67	0x4000_0AC0	DMA_Interrupt	
0x4000_0B68-0x4000_0B97	0x4000_0B68	Soft_HandShake	
0x4000_0B98-0x4000_0BB7	0x4000_0B98	DMA_Control	
0x4000_0BB8-0x4000_0BFF	0x4000_0BB8	预留	

18.8.2 DMAC配置寄存器（DmaCfgReg_H/ DmaCfgReg_L）

- **Name: DMAC Configuration Register**
- **Size: 32 bits**
- **Address Offset: 0x398**
- **Read/Write Access: Read/Write**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
预留															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
预留															DMA_EN

Bit	Name	W/R	Description
63:1	预留	-	预留
0	DMA_EN	W/R	DMA 外设时能 0-DMA 外设关闭 1-DMA 外设打开

18.8.3 DMAC通道使能寄存器（ChEnReg）

- **Name: DW_ahb_dmac Channel Enable Register**
- **Size: 32 bits**
- **Address Offset: 0x3a0**
- **Read/Write Access: Read/Write**

CHx_EN_WE保护各通道使能操作，在对CHx_EN进行写操作同时需要对对应CHx_EN_WE写“1”。
例如：使能通道0，对ChEnReg写0x01x1，其中仅通道0操作有效，其余x对应位操作无效。

31	30	29	28	27	26	25	24
预留							
23	22	21	20	19	18	17	16
预留							
15	14	13	12	11	10	9	8
预留				CH3_EN_WE	CH2_EN_WE	CH1_EN_WE	CH0_EN_WE
7	6	5	4	3	2	1	0
预留				CH3_EN	CH2_EN	CH1_EN	CH0_EN

Bit	Name	W/R	Description
31:12	预留	-	预留
11	CH3_EN_WE	W	DMAC 通道 3 使能写保护位
10	CH2_EN_WE	W	DMAC 通道 2 使能写保护位
9	CH1_EN_WE	W	DMAC 通道 1 使能写保护位
8	CH0_EN_WE	W	DMAC 通道 0 使能写保护位
7:4	预留	-	预留
3	CH3_EN	W/R	DMAC 通道 3 使能位 0-通道关闭 1-通道打开
2	CH2_EN	W/R	DMAC 通道 2 使能位 0-通道关闭 1-通道打开
1	CH1_EN	W/R	DMAC 通道 1 使能位 0-通道关闭 1-通道打开
0	CH0_EN	W/R	DMAC 通道 0 使能位 0-通道关闭 1-通道打开

18.8.4 通道x源地址寄存器（SARx）（x=0...3）

- **Name: Source Address Register for Channel x**
- **Size: 32 bits (upper 32 bits are reserved)**
- **Read/Write Access: Read/Write**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SARx															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SARx															

Bit	Name	W/R	Description
31:0	SARx	W/R	通道 x 源地址寄存器

18.8.5 通道x目标寄存器（DARx）（x=0...3）

- **Name: Destination Address Register for Channel x**
- **Size: 32 bits (upper 32 bits are reserved)**
- **Read/Write Access: Read/Write**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DARx															
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
DARx															

Bit	Name	W/R	Description
31:0	DARx	W/R	通道 x 目标寄存器

18.8.6 通道x链表指针寄存器（LLPx）(x=0...3)

- **Name: Destination Address Register for Channel x**
- **Size: 32 bits (upper 32 bits are reserved)**
- **Address Offset: for x = 0...3:**
  LLP0 – 0x010
  LLP1 – 0x068
  LLP2 – 0x0C0
  LLP3 – 0x118

- **Read/Write Access: Read/Write**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LOC															
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
LOC														预留	

Bit	Name	W/R	Description
31:2	LOC	RW	下一个链表项(LLI)对应的内存起始地址，起始地址的低 2bit 不生效(地址固定为 32bit 对齐)
1:0	预留	RW	保留位，值不可更改

18.8.7 通道x控制寄存器（CTLx_H）(x=0...3)

- **Name: Control Register for Channel x**
- **Size: 64 bits**
- **Address Offset: for x = 0...3:**
  CTL0 – 0x01C
  CTL1 – 0x074
  CTL2 – 0x0CC
  CTL3 – 0x124

- **Read/Write Access: Read/Write**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
预留															
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
预留			DO	BLOCK_TS											

	NE	

Bit	Name	W/R	Description
31:13	预留	-	保留位
12	DONE	W/R	回写功能使能后，块传输完成后该标志置位，同时该寄存器值将被写入链表项对应的控制寄存器 CTL_H。 软件可查询 LLI.CTL_H.Done 标志判断一次块传输是否完成
11:0	BLOCK_TS	W/R	DMA 传输数据块大小，即 XXX_TR_WIDTH 单位数据数量 DMA 传输数据时是以为单位数据，1 次 DMA 传输数据块大小即以 XXX_TR_WIDTH 为单位数据的个数。 启动 DMA 后读取该值表示当前已传输的数据个数 1 次 DMA 传输字节（Byte）量 = BLOCK_TS * XXX_TR_WIDTH / 8

18.8.8 通道x控制寄存器（CTLx_L）(x=0...3)

- **Name: Control Register for Channel x**
- **Size: 64 bits**
- **Address Offset: for x = 0...3:**
  CTL0 – 0x018
  CTL1 – 0x070
  CTL2 – 0x0C8
  CTL3 – 0x120

- **Read/Write Access: Read/Write**

31	30	29	28	27	26	25	24
预留			LLP_SRC_EN	LLP_DST_EN	预留		
23	22	21	20	19	18	17	16
预留	TT_FC			预留			
15	14	13	12	11	10	9	8
SRC_MSIZE		DEST_MSIZE			SINC		
7	6	5	4	3	2	1	0
DINC	SRC_TR_WIDTH			DST_TR_WIDTH			INT_EN

Bit	Name	W/R	Description
31:29	预留	RO	保留位
28	LLP_SRC_EN	W/R	多块链源端使能位 1：使能
27	LLP_DST_EN	W/R	多块链目的端使能位 1：使能
26:23	预留	RO	保留位
22:20	TT_FC	W/R	传输方式及流控制器选择 设定值与传输方式对应表： 设定值 | 传输方式

<table>
<tr><td></td><td></td><td></td><td><table><tr><td>000</td><td>Memory to Memory</td></tr><tr><td>001</td><td>Memory to Peripheral</td></tr><tr><td>010</td><td>Peripheral to Memory</td></tr></table></td></tr>
<tr><td>19:17</td><td>预留</td><td>-</td><td></td></tr>
<tr><td>16:14</td><td>SRC_MSIZE</td><td>W/R</td><td>源数据突发传输量（Source Burst Transaction Length）
源数据 1 次突发传输中，传输 SRC_TR_WIDTH 的数量，即传输多少个 SRC_TR_WIDTH。
源数据 1 次突发传输 Byte = SRC_MSIZE * SRC_TR_WIDTH / 8
设定值与数据量对应表：<table><tr><td>设定值</td><td>数据量</td></tr><tr><td>000</td><td>1</td></tr><tr><td>001</td><td>4</td></tr><tr><td>010</td><td>8</td></tr><tr><td>011</td><td>16</td></tr><tr><td>100</td><td>32</td></tr><tr><td>101</td><td>64</td></tr><tr><td>110</td><td>128</td></tr><tr><td>111</td><td>256</td></tr></table></td></tr>
<tr><td>13:11</td><td>DEST_MSIZE</td><td>W/R</td><td>目标数据突发传输量（Destination Burst Transaction Length）
源数据 1 次突发传输中，传输 DST_TR_WIDTH 的数量，即传输多少个 DST_TR_WIDTH。
目标数据 1 次突发传输 Byte = DEST_MSIZE * DST_TR_WIDTH / 8
设定值与数据量对应表同 SRC_MSIZE</td></tr>
<tr><td>10:9</td><td>SINC</td><td>W/R</td><td>源地址增量模式
00 = 地址自加
01 = 地址自减
1x = 地址不变</td></tr>
<tr><td>8:7</td><td>DINC</td><td>W/R</td><td>目标地址增量模式
00 = 地址自加
01 = 地址自减
1x = 地址不变</td></tr>
<tr><td>6:4</td><td>SRC_TR_WIDTH</td><td>W/R</td><td>源数据位宽

设定值与数据位宽对应表：<table><tr><td>设定值</td><td>数据位宽</td></tr><tr><td>000</td><td>8</td></tr><tr><td>001</td><td>16</td></tr><tr><td>010</td><td>32</td></tr><tr><td>011</td><td>64</td></tr><tr><td>100</td><td>128</td></tr><tr><td>101</td><td>256</td></tr><tr><td>11x</td><td>256</td></tr></table></td></tr>
<tr><td>3:1</td><td>DST_TR_WIDTH</td><td>W/R</td><td>目标数据位宽

设定值与数据位宽对应同 SRC_TR_WIDTH</td></tr>
<tr><td>0</td><td>INT_EN</td><td>W/R</td><td>DMA 中断总控制位
0-关闭 DMA 所有中断
1-打开 DMA 所有中断</td></tr>
</table>

18.8.9 通道x配置寄存器（CFGx_L）(x=0...3)

- **Name: Configuration Register for Channel x**
- **Size: 32 bits (upper 32 bits are reserved)**
- **Address Offset: for x = 0:**
  CFG0 – 0x040
  CFG1 – 0x098
  CFG2 – 0x0F0
  CFG3 – 0x148

- **Read/Write Access: Read/Write**

31	30	29	28	27	26	25	24
RELOAD_DST	RELOAD_SRC	预留					

23	22	21	20	19	18	17	16
预留				SRC_HS_POL	DST_HS_POL	预留	

15	14	13	12	11	10	9	8
预留				HS_SEL_SRC	HS_SEL_DST	FIFO_EMPTY	CH_SUSP

7	6	5	4	3	2	1	0
CH_PRIOR			预留				

Bit	Name	W/R	Description
31	RELOAD_DST	W/R	自动加载目的地址 多块传输模式下，每块传输完成后 DARx 寄存器自动加载其初始值，一次新的块传输即被初始化 1：使能；0：不使能
30	RELOAD_SRC	W/R	自动加载源地址 多块传输模式下，每块传输完成后 SARx 寄存器自动加载其初始值，一次新的块传输即被初始化 1：使能；0：不使能
29:20	预留	-	保留位
19	SRC_HS_POL	W/R	源设备握手信号有效极性 0-高有效 1-低有效
18	DST_HS_POL	W/R	目标设备握手信号有效极性 0-高有效 1-低有效
17:12	预留	-	保留位
11	HS_SEL_SRC	W/R	源设备软/硬握手接口选择 0-硬握手 1-软握手
10	HS_SEL_DST	W/R	目标设备软/硬握手接口选择 0-硬握手 1-软握手
9	FIFO_EMPTY	R	通道 FIFO 状态

			0-通道 FIFO 未满 1-通道 FIFO 满
8	CH_SUSP	W/R	通道挂起（Channel Suspend） 0-不挂起 1-挂起 DMA 通道从源设备获取数据
7:5	CH_PRIOR	W/R	通道优先级 设定值范围： 0 < CH_PRIOR < 7(通道数 - 1) 超出设定范围会导致异常错误
4:0	预留	-	保留位

18.8.10源中断状态寄存器

- **Name: Interrupt Raw Status Registers**
- **Size: 32 bits**
- **Address Offset:**
  RawTfr – 0x2c0
  RawBlock – 0x2c8
  RawSrcTran – 0x2d0
  RawDstTran – 0x2d8
  RawErr – 0x2e0
- **Read/Write Access: Read/Write**

原中断状态寄存器包括：RawBlock、RawDstTran、RawErr、RawSrcTran、RawTfr
具体说明见“DMA中断”章节

以上寄存器结构及对应通道相同，操作地址不同。
原中断状态寄存器中状态值不受中断屏蔽寄存器（Mask）影响。
通过对对应中断清除寄存器写“1”，清除相应原中断状态寄存器中的标记位。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
预留															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
预留												CH 3	CH 2	CH 1	CH 0

Bit	Name	W/R	Description
31:4	预留	-	预留
3	CH3	W/R	源中断标志位，写操作通常用于测试，普通模式不建议使用写操作。 0-未产生中断 1-产生中断
2	CH2	W/R	
1	CH1	W/R	
0	CH0	W/R	

18.8.11中断状态寄存器

- **Name: Interrupt Status Registers**
- **Size: 32 bits**
- **Address Offset:**
  StatusTfr – 0x2c0
  StatusBlock – 0x2c8
  StatusSrcTran – 0x2d0
  StatusDstTran – 0x2d8

StatusErr – 0x2e0
- **Read/Write Access: Read/Write**

中断状态寄存器包括：StatusBlock、StatusDstTran、StatusErr、StatusSrcTran、StatusTfr
具体说明见“DMA中断”章节

以上寄存器结构及对应通道相同，操作地址不同。
中断状态寄存器中状态值会中断屏蔽寄存器（Mask）影响，当中断状态寄存器bit位置“1”，外设会对CPU产生中断信号。
通过对对应中断清除寄存器写“1”，清除相应原中断状态寄存器中的标记位。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
预留															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
预留												CH3	CH2	CH1	CH0

Bit	Name	W/R	Description
31:4	预留	-	
3	CH3	R	中断状态标志位。 0-未产生中断 1-产生中断
2	CH2	R	
1	CH1	R	
0	CH0	R	

18.8.12中断屏蔽寄存器

- **Name: Interrupt Mask Registers**
- **Size: 32 bits**
- **Address Offset:**
  MaskTfr – 0x310
  MaskBlock – 0x318
  MaskSrcTran – 0x320
  MaskDstTran – 0x328
  MaskErr – 0x330
- **Read/Write Access: Read/Write**

中断屏蔽寄存器包括：MaskBlock、MaskDstTran、MaskErr、MaskSrcTran、MaskTfr
具体说明见“DMA中断”章节

以上寄存器结构及对应通道相同，操作地址不同。
中断屏蔽寄存器（Mask）用于屏蔽DMA中断，影响中断状态寄存器（**Status**）值。
每个中断屏蔽位对应1个写保护操作为。
例如：打开通道0中断，对**MaskBlock**写0x01x1，其中仅通道0操作有效，其余x对应位操作无效。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
预留															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
预留				CH3_WE	CH2_WE	CH1_WE	CH0_WE	预留				CH3	CH2	CH1	CH0

Bit	Name	W/R	Description
31:12	预留	-	

11	CH3_WE	W	中断屏蔽写保护位 0-CHx 写操作无效 1-CHx写操作有效
10	CH2_WE	W	
9	CH1_WE	W	
8	CH0_WE	W	
7:4	预留	-	
3	CH3	W/R	中断状态标志位。 0-中断关闭 1-中断打开
2	CH2	W/R	
1	CH1	W/R	
0	CH0	W/R	

18.8.13中断状态寄存器

- **Name: Interrupt Clear Registers**
- **Size: 32 bits**
- **Address Offset:**
  ClearTfr – 0x2c0
  ClearBlock – 0x2c8
  ClearSrcTran – 0x2d0
  ClearDstTran – 0x2d8
  ClearErr – 0x2e0
- **Read/Write Access: Read/Write**

中断状态寄存器包括：ClearBlock、ClearDstTran、ClearErr、ClearSrcTran、ClearTfr
具体说明见“DMA中断”章节

以上寄存器结构及对应通道相同，操作地址不同。
对各中断清除寄存器写“1”，原中断寄存器中的标志位会在同一周期被清“0”。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
预留															
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
预留												CH3	CH2	CH1	CH0

Bit	Name	W/R	Description
31:4	预留	-	
3	CH3	W	中断状态标志位。 0-无效 1-清除中断
2	CH2	W	
1	CH1	W	
0	CH0	W	

18.8.14组合中断状态寄存器（ChEnReg）

- **Name: Combined Interrupt Status Register**
- **Size: 32 bits**
- **Address Offset: 0x360**
- **Read/Write Access: Read/Write**

以下中断状态位各中断状态寄存器状态值或运算后结果，即4个通道各中断状态或运算结果。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16

预留															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
预留											ERR	DSTT	SRCT	BLOCK	TFR

Bit	Name	W/R	Description
31:5	预留	-	
4	ERR	R	错误状态中断
3	DSTT	R	目标数据传输完成中断
2	SRCT	R	源数据传输完成中断
1	BLOCK	R	块传输中断
0	TFR	R	传输完成中断

18.8.15软握手接口寄存器

- **Name: Interrupt Mask Registers**
- **Size: 32 bits**
- **Address Offset:**
  ReqSrcReg– 0x368
  ReqDstReg–0x370
  SglReqSrcReg–0x378
  SglReqDstReg–0x380
  LstSrcReg–0x388
  LstDstReg–0x390
- **Read/Write Access: Read/Write**

软握手接口寄存器：ReqSrcReg、ReqDstReg、SglReqSrcReg、SglReqDstReg、LstSrcReg、LstDstReg功能说明见“DMA软握手接口”章节

以上寄存器结构及对应通道相同，操作地址不同。
每个中断屏蔽位对应1个写保护操作为。
例如：对应通道0产生源数据DMA请求，对**ReqSrcReg**写0x0101，其中仅通道0操作有效，其余x对应位操作无效。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
预留															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
预留				CH3_WE	CH2_WE	CH1_WE	CH0_WE	预留				CH3	CH2	CH1	CH0

Bit	Name	W/R	Description
31:12	预留	-	
11	CH3_WE	W	DMA 请求写保护位 0-CHx 写操作无效 1-CHx写操作有效
10	CH2_WE	W	
9	CH1_WE	W	
8	CH0_WE	W	
7:4	预留	-	
3	CH3	W/R	DMA 请求位。 0-无效 1-产生请求
2	CH2	W/R	
1	CH1	W/R	
0	CH0	W/R	

19 USB接口

19.1 USB简介

USB外设实现了USB2.0全速总线和APB1总线间的接口
USB外设支持USB挂起/恢复操作，可以停止设备时钟实现低功耗
USB2.0 OTG，支持低速和全速模式（8 endpoints）

19.2 USB主要特点

- 符合USB2.0全速设备的技术规范
- CRC（循环冗余校验）生成/校验，反向不归零（NRZI）编码/解码和位填充

注：USB和CAN公用一个专用的512字节的SRAM存储器用于数据的发送和接收，因此不能同时使用USB和CAN（共享的SRAM被CAN模块互斥地访问），USB和CAN可以同时用于一个应用中但不能在同一时间使用。

19.3 USB功能描述

USB模块为PC主机和微控制器所实现的功能之间提供了符合USB规范的通信连接。PC主机和微控制器之间的数据传输是通过共享一专用的数据缓冲区来完成的，该数据缓冲区能被USB外设直接访问。

19.4 USB寄存器描述

USB模块的寄存器有以下三类：

- 通用类寄存器：中断寄存器和控制寄存器
- 端点类寄存器：端点配置寄存器和状态寄存器
- 缓冲区描述表类寄存器：用来确定数据分组存放地址的寄存器

缓冲区描述表类寄存器的基地址由USB_BTABLE寄存器指定，所有其他寄存器的基地址则为USB模块的基地址0x4000 5C00。同样的地址对齐方式也用于从0x4000 6000开始的分组缓冲存储区。

19.4.1 地址映射表

USB基地址列表

地址范围	基地址	外设	总线
0x4000_0C00-0x4000_0FFF	0x4000_0C00	USB	AHB

表 19-1 USB寄存器表

偏移地址	寄存器名称	宽度（bit）	复位值
0x00	FADDR	8	0x00
0x01	POWER	8	0x20
0x02	INTRTX	16	0x0000
0x04	INTRRX	16	0x0000
0x06	INTRTXE	16	0xFFFF
0x08	INTRRXE	16	0xFFFE
0x0A	INTRUSB	8	0x00
0x0B	INTRUSBE	8	0x06
0x0C	FRAME	16	0x0000
0x0E	INDEX	8	0x00
0x0F	TESTMODE	8	0x00
0x10	TXMAXP	16	0x0000

0x12	TXCSRL	8	0x00
0x13	TXCSRH	8	0x00
0x14	RXMAXP	16	0x0000
0x16	RXCSRL	8	0x00
0x17	RXCSRH	8	0x00
0x18	RXCOUNT	16	0x0000
0x1A	TXTYPE	8	0x00
0x1B	TXINTERVAL	8	0x00
0x1C	RXTYPE	8	0x00
0x1D	RXINTERVAL	8	0x00
0x1F	FIFOSIZE	8	Configuration Dependent
0x20-0x5F	FIFOx	32	0x00000000
0x60	DEVCTL	8	0x80
0x61	MISC	8	0x00
0x62	TXFIFOSZ	8	0x00
0x63	RXFIFOSZ	8	0x00
0x64	TXFIFOADD	16	0x0000
0x66	RXFIFOADD	16	0x0000
0x68	VCONTROL/VSTATUS	32	-
0x6C	HWVERS	32	Version dependent
0x70-0x77	-	-	-
0x78	EPINFO	8	Implementation dependent
0x79	RAMINFO	8	Implementation dependent
0x7A	LINKINFO	8	0x5C
0x7B	VPLEN	8	0x3C
0x7C	HS_EOF1	8	0x80
0x7D	FS_EOF1	8	0x77
0x7E	LS_EOF1	8	0x72
0x7F	SOFT_RST	8	0x00
0x80+8*n	TXFuncAddr	8	0x00
0x82+8*n	TXHubAddr	8	0x00
0x83+8*n	TXHubPort	8	0x00
0x84+8*n	RxFuncAddr	8	0x00
0x86+8*n	RxHubAddr	8	0x00
0x87+8*n	RxHubPort	8	0x00
0x100+16*n	TXMAXP	16	0x0000
0x102+16*n	TXCSRL	8	0x00
0x103+16*n	TXCSRH	8	0x00
0x104+16*n	RXMAXP	16	0x0000
0x106+16*n	RXCSRL	8	0x00
0x107+16*n	RXCSRH	8	0x00
0x108+16*n	RXCOUNT	16	0x0000
0x10A+16*n	TXTYPE	8	0x00
0x10B+16*n	TXINTERVAL	8	0x00
0x10C+16*n	RXTYPE	8	0x00
0x10D+16*n	RXINTERVAL	8	0x00
0x10F+16*n	FIFOSIZE	8	Configuration Dependent
0x200	DMA_INTR	8	0x00

0x204	DMA_CNTL	16	0x0000
0x208	DMA_ADDR	32	0x00000000
0x20C	DMA_COUNT	32	0x00000000
0x300+2*n	RqPktCount	16	0x0000
0x340	RxDPktBufDis	16	0x0000
0x342	TxDPktBufDis	16	0x0000
0x344	C_T_UCH	16	Various
0x346	C_T_HSRTN	16	Various
0x348	C_T_HSBT	16	0x0000
0x360	LPM_ATTR	16	0x00
0x362	LPM_CNTRL	8	0x00
0x363	LPM_INTREN	8	0x00
0x364	LPM_INTR	8	0x00
0x365	LPM_FADDR	8	0x00

19.4.2 USB 通用寄存器(Common Registers)

FADDR

偏移地址：0x0000　　　　复位值：0x00

注：仅在外设模式下有效

D7	D6	D5	D4	D3	D2	D1	D0
0	Function Address						
r	rw						

Bit	Name	W/R	Description
D7	-	-	未用，读取一直返回 0
D6:D0	Func Addr	W/R	当用于设备模式(DevCtl.D2=0),写入通过 SET_ADDRESS 命令接收到的地址，随后的 Token 包将会自动使用该地址。

POWER

偏移地址：0x0001　　　　复位值：0x20

	D7	D6	D5	D4	D3	D2	D1	D0
	ISO Update	Soft Conn	HS Enab	HS Mode	Reset	Resume	Suspend Mode	Enable SuspendM
Peripera	rw	rw	rw	r	r	rw	r	rw
Host	-	-	rw	r	rw	rw	w	rw

Bit	Name	W/R	Description
D7	ISO Update	W/R	设置此位，将会从设置 TxPktRdy 时开始等待接收一个 SOF Token 后发送数据包。如果在等待 SOF Token 期间先接收到一个 IN Token，将会发送一个 0 长度的数据包。 **注：该位仅用于设备模式下有效。并且仅适用与等时传输(Isocbronous transfers)**
D6	Soft Conn	W/R	当使能软件连接/断开功能，设置此位使能 USB D+/D-线，清除此位 USB D+/D-线处于尝试状态。 **注：该位仅用于设备模式下有效。**

D5	HS Enab	W/R	暂不支持
D4	HS Mode	R	暂不支持
D3	Reset	Host:W/R Peripheral mode:RO	当检测到总线上有 Reset 信号时改为置 1。 **注：在 Host Mode 下改为可读可写；在 Peripheral Mode 下只读。**
D2	Resume	W/R	当设备处于 Suspend 模式时，置位此位将会产生 Resume 信号。在 Peripheral Mode 下应该在 10ms(最大 15ms)后清除该位；在 Host Mode 下应该在 20ms 后清除该位。
D1	Suspend Mode	R	在 Host Mode 下，置位此位进入 Suspend Mode；在 Peripheral Mode 下，进入 Suspend Mode 将会设置此位。读取中断寄存器或者置位上面的 Resume 位将会清除该位。
D0	Enable SuspendM	W/R	置位此位使能 SUSPENDM 输出。

INTRTX

偏移地址：0x0002 复位值：0x0000

IntrTx 是一个16位的只读寄存器用于指示端点0和Tx端点1-3当前发生的中断。

注：对没有配置的端点中断位将会一直为0。同时对该寄存器进行读操作将会清除所有有效的中断指示位。

D15	D14	D13	D12	D11	D10	D9	D8
预留							
ro							

D7	D6	D5	D4	D3	D2	D1	D0
预留				EP3 Tx	EP2 Tx	EP1 Tx	EP0
ro				r	r	r	r

Bit	Name	W/R	Description
D15-D4	预留	RO	预留
D3	EP3 Tx	RO	Tx 端点 3 中断
D2	EP2 Tx	RO	Tx 端点 2 中断
D1	EP1 Tx	RO	Tx 端点 1 中断
D0	EP0	RO	端点 0 中断

INTRRX

偏移地址：0x0004 复位值：0x0000

IntrRx 是一个16位的只读寄存器用于指示Rx端点1-3当前发生的中断。

注：对没有配置的端点中断位将会一直为0。同时对该寄存器进行读操作将会清除所有有效的中断指示位。

D15	D14	D13	D12	D11	D10	D9	D8
预留							
ro							

D7	D6	D5	D4	D3	D2	D1	D0
预留				EP3 Rx	EP2 Rx	EP1 Rx	-
ro				r	r	r	r

Bit	Name	W/R	Description
D15-D4	预留	RO	预留

D3	EP3 Rx	RO	Rx 端点 3 中断
D2	EP2 Rx	RO	Rx 端点 2 中断
D1	EP1 Rx	RO	Rx 端点 1 中断
D0	-	RO	未使用

INTRTXE

偏移地址：0x0006 复位值：0xFFFF

IntrTxE 是一个为IntrTx寄存器提供中断使能的16位的寄存器。对应的Bit位写1将使能对应的中断，当中断事件发生时将会置位IntrTx对应的Bit位并触发中断，当对应的Bit位为0时当中断事件发生时也会置位intrTx对应的Bit位但是不会触发中断。

注：对没有配置的位将会一直为0。

D15	D14	D13	D12	D11	D10	D9	D8
预留							
ro							

D7	D6	D5	D4	D3	D2	D1	D0
预留				EP3 Tx	EP2 Tx	EP1 Tx	EP0
ro				rw	rw	rw	rw

Bit	Name	W/R	Description
D15-D4	预留	RO	预留
D3	EP3 Tx	W/R	Tx 端点 3 中断使能
D2	EP2 Tx	W/R	Tx 端点 2 中断使能
D1	EP1 Tx	W/R	Tx 端点 1 中断使能
D0	EP0	W/R	端点 0 中断使能

INTRRXE

偏移地址：0x0008 复位值：0xFFFF

IntrRxE 是一个为IntrRx寄存器提供中断使能的16位的寄存器。对应的Bit位写1将使能对应的中断，当中断事件发生时将会置位IntrRx对应的Bit位并触发中断，当对应的Bit位为0时当中断事件发生时也会置位intrRx对应的Bit位但是不会触发中断。

注：没有配置的位将一直为0。

D15	D14	D13	D12	D11	D10	D9	D8
预留							
ro							

D7	D6	D5	D4	D3	D2	D1	D0
				EP3 Tx	EP2 Tx	EP1 Tx	-
ro				rw	rw	rw	r

Bit	Name	W/R	Description
D15-D4	预留	RO	预留
D3	EP3 Rx	W/R	Rx 端点 3 中断使能
D2	EP2 Rx	W/R	Rx 端点 2 中断使能
D1	EP1 Rx	W/R	Rx 端点 1 中断使能
D0	-	RO	未使用

INTRUSB

偏移地址：0x000A 复位值：0x00

D7	D6	D5	D4	D3	D2	D1	D0
VBus Error	Sess Req	Discon	Conn	SOF	Reset/ Babble	Resume	Suspend
ro	ro	ro	ro	ro	ro	ro	ro

Bit	Name	W/R	Description
D7	VBus Error	R	在会话过程中，当 VBus 低于 VBus 有效门限值时置位。**仅在作为'A'设备时有效。**
D6	Sess Req	R	检测到 Session Request 信号时置位。**仅在作为'A'设备时有效。**
D5	Discon	R	在 Host Mode 下检测到设备断开连接时置位；在 Peripheral Mode 下一个会话(session)结束时置位。在所有类型传输速度下均有效。
D4	Conn	R	当检测到设备连接时置位。**仅在 Host Mode 模式下有效，**所有类型传输速度下均有效。
D3	SOF	R	当一个新帧(Frame)开始时置位
D2	Reset	R	在 Peripheral Mode 下当在总线上检测到 Reset 信号时置位。
	Babble	R	在 Host Mode 下检测到 Babble 时置位。 **注：仅在第一个 SOF 发送之后启动(active).**
D1	Resume	R	在 Suspend Mode 下。检测到总线上有 Resume 信号置位。
D0	Suspend	R	检测到总线上有 Suspend 信号时置位。 注：仅在 Peripheral Mode 有效。

INTRUSBE

偏移地址：0x000B　　　　复位值：0x06

IntrUSBE是一个为IntrUSB中每个中断提供中断使能的8位寄存器。

D7	D6	D5	D4	D3	D2	D1	D0
VBus Error	Sess Req	Discon	Conn	SOF	Reset /Babble	Resume	Suspend
rw	rw	rw	rw	rw	rw	rw	rw

FRAME

偏移地址：0x000C　　　　复位值：0x0000

Frame是一个用于保存最新接收的帧号(Frame number)的16位只读寄存器。

D15	…			D11	D10 … D0
0	0	0	0	0	(MSB) Frame Number (LSB)
r	…			r	r … r

INDEX

偏移地址：0x000E　　　　复位值：0x00

每一个TX端点，每个接收终端都有自己的一套位于100H-1FFh之间的控制/状态寄存器。另外一组TX控制/状态和一组接收控制/状态和一组接收控制/状态寄存器出现在10H-19H。

Index是一个四位寄存器，用于确定哪些端点控制/状态寄存器被访问。

D3	D2	D1	D0
(MSB)	Selected Endpoint		(LSB)
rw	rw	rw	rw

在访问端点控制/状态寄存器的10h-19h之前，端点数量应写入变址寄存器，以确保正确的控制/状态寄存器会出现内存映射。

TESTMODE

偏移地址： 0x0F 复位值：0x00

TESTMODE是一个8位寄存器，主要把MH1902T应用到USB2.0规范中描述的四种测试模式中用于高速运行的一个

D7	D6	D5	D4	D3	D2	D1	D0
Force_Host	FIFO_Access	Force_FS	Force_HS	Test_Packet	Test_K	Test_J	Test_SE0_NAK
rw	w	rw	rw	rw	rw	rw	rw

Bit	Name	W/R	Description
D7	Force_Host	W/R	CPU 设置该位指示核心进入主机模式时，会话位设置，不管其是否连接到外设。在 CID 输入状态下，HostDisconnect 和 LineState 信号被忽略。然后该核心将保持在主机模式下，直到会话位被清零，即使设备断开，如果 Force_Host 位保持设置，将重新进入主机模式下一次会话位被设置，在此模式下，从 PHY 的 HOSTDISCON 信号的状态可以从位 7 读该 DevCtl 寄存器，运作速度从 Force_FS 位确定如下： Force_FS / 运行速度 0 / 低速 1 / 全速 1 / 未定义
D6	FIFO_Access	W	CPU 设置该位为端点 0 Tx FIFO 传送数据包发送到端点 0 接收 FIFO，它会被自动清除
D5	Porce_FS	W/R	CPU 设置该位或与 7 位以上或强制 MH1902T 一起全速接收到一个 USB 复位时的模式
D4	Porce_HS	W/R	CPU 设置该位或与 7 位以上或强制 MH1902T 一起高速接收到一个 USB 复位时的模式
D3	Test_Packet	W/R	CPU 设置该位进入 Test Packet 测试模式。在这种模式下，MH1902T 反复发送该总线上的 53 字节的测试数据包，其中的形式被定义在通用串行总线规范 2.0 修订版，第 7.1.20 **注：测试包有一个固定的格式，必须被加载到端点 0 FIFO 测试模式之前被输入**
D2	Test_K	W/R	CPU 设置该位进入 Test_K 测试模式，在这种模式下，MH1902T 在总线上连续发送 K
D1	Test_J	W/R	CPU 设置该位进入 Test_J 测试模式，在这种模式下，MH1902T 在总线上连续发送 J
D0	Test_sE0_NAK	W/R	CPU 设置该位进入 Test_SE0_NAK 测试模式，在这种模式下，MH1902T 保持在高速模式，但响应任何带有 NAK 的有效 IN 令牌。

DEVCTL

偏移地址： 0x60 复位值：0x80

DevCtl是用于选择在外设模式或在主机模式下的MH1902T是否运行一个8位寄存器和为控制监视USB VBUS线路。如果所述PHY被挂起，则没有PHY时钟（XCLK）被接收，VBUS不采样

D7	D6	D5	D4	D3	D2	D1	D0
B-Device	FSDev	LSDev	VBus[1]	VBus[0]	Host Mode	Host Req	Session
r	r	r	r	r	r	rw	rw

<table>
<tr><th>Bit</th><th>Name</th><th>W/R</th><th>Description</th></tr>
<tr><td>D7</td><td>B-Device</td><td>W/R</td><td>该只读位指示 MH1902T 是否运行为“A”设备或“B”装置，0=>’A’装置，1=>’B’设备。只有有效时会话进行，当没有会话进行时确定角色，设置会话位并读取该位。
注意：如果核心是在 Force_Host 模式（即会话已经开始且 Testmode.D7=1），该位将指示从 PHY 所述的 HOSTDISCON 输入信号状态</td></tr>
<tr><td>D6</td><td>FSDev</td><td>R</td><td>这个只读位被设置时，全速装置被检测到连接到该端口。只有在主机模式下有效</td></tr>
<tr><td>D5</td><td>LSDev</td><td>R</td><td>这个只读位被设置时，低速装置被检测到连接到该端口。只有在主机模式下有效</td></tr>
<tr><td>D4-D3</td><td>VBus[1:0]</td><td>R</td><td>HESE 只读位如下编码当前 VBUS 级别：
<table>
<tr><td>D4</td><td>D3</td><td>Meaning</td></tr>
<tr><td>0</td><td>0</td><td>低于 SESSIOONEND</td></tr>
<tr><td>0</td><td>1</td><td>SESSIONEND 以上,AValid 以下</td></tr>
<tr><td>1</td><td>0</td><td>AValid 以上，VBusValid 以下</td></tr>
<tr><td>1</td><td>1</td><td>VBusValid 以上</td></tr>
</table></td></tr>
<tr><td>D2</td><td>Host Mode</td><td>R</td><td>这个只读位在 MH1902T 充当主机设置</td></tr>
<tr><td>D1</td><td>Host Req</td><td>W/R</td><td>当设置时，MH1902T 将启动主机协商时输入挂起模式。当主机协商完成后清零。</td></tr>
<tr><td>D0</td><td>Session</td><td>W/R</td><td>当作为’A’装置工作时，该位被设置或者由 CPU 清零开始或结束会话。当作为一个’B’装置工作时，该位被置位，由当会话开始/结束的 MH1902T 清除。它也由 CPU1 以启动会话请求协议。当 MH1902T 是暂停模式时，该位可以由 CPU 清除为执行软件断开
注：当核心未被暂停时清除该位将导致不确定的行为发生</td></tr>
</table>

MISC

偏移地址：0x60 复位值：0x00

MISC寄存器是包含各种常见的配置位的8位寄存器，这些位包括RX/TX早于DMA的使能位。

D7	D6	D5	D4	D3	D2	D1	D0
Unused						tx_edma	rx_edma
r	r	r	r	r	r	rw	rw

Bit	Name	W/R	Description
D7-D2	Unused	R	这些位预留
D1	tx_edma	W/R	1’b0：DMA_REQ信号，所有IN端点将被重新拉高时MAXP字节都被写入到一个端点，这是一个后期的模式。 1’b1：DMA_REQ信号，所有IN端点将被重新拉高时MAXP-8字节已经写入到端点，这是早期的模式。
D0	rx_edma	W/R	1’b0：DMA_REQ信号，所有OUT端点将被重新拉高时MAXP字节读到端点，这是一个后期的模式。

			1’b1：DMA_REQ信号，所有OUT端点将被重新拉高时MAXP-8字节被读取到端点，这是早期的模式。

19.4.3 USB 索引寄存器(Indexed registers)

CSR0L

CSR0L是一个8位寄存器，用于提供控制和状态位端点0。

注：该寄存器的解释取决于MH1902T是否充当外设或作为主机。用户也应该注意当读取寄存器反映状态作为写入寄存器的结果时的返回值。

CSROL在外设模式下：

D7	D6
Serviced SetupEnd (self-clearing)	Serviced RxPktRdy (self-clearing)
w	w

D5	D4	D3	D2	D1	D0
SendStall (self-clearing)	SetupEnd	DataEnd (self-clearing)	SentStall	TxPktRdy (self-clearing)	RxPktRdy
w	r	w	rw	rw	r

Bit	Name	W/R	Description
D7	ServicedSetupEnd	W	该CPU写1到该位，清除SetupEnd位。它会自动清零。
D6	ServicedRxPktRdy	W	该CPU写1到该位，清除RxPktRdy位。它会自动清零。
D5	SendStall	W	该CPU写1到该位来终止当前事务。STALL握手会被发送，然后该位将会被自动清除
D4	SetupEnd	R	控制交易前DATAEND位已经设置结束，该位将被设置。此时将产生中断，FIFO刷新。该位由CPU写1到SetupEnd位清零。
D3	DataEnd	W	CPU设置此位： 1． 当设置TxPktRdy为最后的数据包时 2． 当卸载最后一个数据包后清零RxPktRdy 3． 在设定为TxPktRdy零长度数据分组 它会自动清零。
D2	SentStall	W/R	当STALL握手发送时设置此位，该CPU应该清除此位
D1	TxPktRdy	W/R	该CPU加载数据包到FIFO后，设置此位，它自动清除时一个数据包被发送，此时产生中断（如果使能）
D0	RxPktRdy	R	该位在数据分组接收时被设置，当此位被设置时产生一个中断。CPU通过设置ServicedRxPktRdy位来清除该位

CSROL 在主机模式下：

D7	D6	D5	D4	D3	D2	D1	D0
NAK Timeout	StatusPkt	ReqPkt	Error	SetupPkt	RxStall	TxPktRdy	RxPktRdy
rc	rw	rw	rc	rc	rc	rw	rc

Bit	Name	W/R	Description
D7	NAK Timeout	C/R	当端点0在收到NAK响应比NAKLimit0寄存器设定的时间长时该位被设置。CPU应清除此位以允许端点继续
D6	StatusPkt	W/R	CPU将该位在同一时间作为TxPktRdy 或ReqPkt位设置，执行状态阶段事务。设置该位保证了数据切换被设置为1，使得DATA1包被用于状态阶段事务

D5	ReqPkt	W/R	CPU设置该位以请求IN事务，当RxPktRdy设置时该位清零
D4	Error	C/R	党已经进行了三次尝试，没有来自周边的响应执行事务，该位将被设置，此时会产生一个中断。该CPU应该清除此位。
D3	SetupPkt	C/R	CPU设置该位作为TxPktRdy位的同时，发送一个交易的SETUP令牌来代替OUT令牌。**注：数据切换时该位也会清除。**
D2	RxStall	C/R	当接收到STALL握手时，该位被设置。该CPU应该清除此位
D1	TxPktRdy	W/R	该CPU加载数据包到FIFO后设置此位，当数据包发送后它会自动清零，此时产生中断（如果使能）。
D0	RxPktRdy	C/R	该位在数据分组接收时被设置，此时产生一个中断（如果使能）。当数据包在FIFO中被读取时CPU清除此位

CSR0H

CSR0H是一个8位寄存器，用于提供控制和状态位端点0。
注：该寄存器的解释取决于MH1902T是否充当外设或作为主机。用户也应该注意当读取寄存器反映状态作为写入寄存器的结果时的返回值。
CSROL 在外设模式下：

D7	D6	D5	D4	D3	D2	D1	D0
-	-	-	-	-	-	-	FlushFIFO (self_clearing)
r	r	r	r	r	r	r	w

Bit	Name	W/R	Description
D7-D1	-	R	未使用，读取时返回0
D0	FlushFIFO	W	该CPU写1到该位来缓冲下一个要发送/从读端点0 FIFO开始读的数据包。FIFO指针被复位，TxPktRdy/ RxPktRdy位（及以下）被清除。**注：当TxPktRdy/ RxPktRdy位被设置时，FlushFIFO才会被使用，在其他时候，这可能会导致数据被破坏**

CSROL 在主机模式下：

D7	D6	D5	D4	D3	D2	D1	D0
-	-	-	-	Dis Ping	Data Toggle Wr.Enable (self-clearing)	Data Toggle	FlushFIFO (self-clearing)
r	r	r	r	rw	w	rw	w

Bit	Name	W/R	Description
D7-D4	-	R	未使用，读取时返回0
D3	Dis Ping	W/R	该CPU写1到该位会显示出核心，不会在数据和状态高速控制转移时发出PING指令
D2	Data Toggle Wr.Enable	W	该CPU写1到该位，会触发结束位为0的数据状态而记录下来（见数据位就触发，下同）。一旦新的值被写入，该位会自动清零
D1	Data Toggle	W/R	读取时，该位指示端点0数据会触发。如果D10为高，该位随着数据触发所需的设置可能被写入。如果D10低，写入该位的任何值将被忽略
D0	FlushFIFO	W	该CPU写1到该位来刷新下一个数据包从端点0 FIFO发送/读取。FIFO指针被复位，TxPktRdy/ RxPktRdy位（以下）被清除 **注：当TxPktRdy/ RxPktRdy位被设置时，FlushFIFO是**

			唯一被使用的，在其他时候，可能会导致数据被破坏

COUNT0

COUNT0是一个7位只读寄存器，用于指示在端点0 FIFO中接收到的数据字节数。当RxPktRdy（CSR0.DO）被设置时，返回的值的变化作为在FIFO变化的内容。

D6	D5	D4	D3	D2	D1	D0
(MSB)			Endpoint 0 Rx Count			(LSB)
r	r	r	r	r	r	r

TYPE0

注：HOST模式下！

D7	D6
Speed	
rw	rw

Bit	Name	W/R	Description
D7-D6	Speed	W/R	目标设备的操作速度： 00：未使用（**注：如果选中，则目标将被认为是使用相同的连接速度为核心**） 10：全 11：低

CONFIGDATA

CONFIGDATA是一个8位只读寄存器，返回所选核心配置的有关信息。

D7	D6	D5	D4	D3	D2	D1	D0
MPRxE	MPTxE	BigEndian	HBRxE	HBTxE	DynFIFO Sizing	SoftConE	UTMI DataWidth
r	r	r	r	r	r	r	r

Bit	Name	W/R	Description
D7	MPRxE	R	当设置为“1”，自动合并的散包被选择
D6	MPTxE	R	当设置为“1”，自动分裂的散包被选择
D5	BigEndian	R	始终为0，表示小端排序
D4	HBRxE	R	当设置位“1”表示高带宽接收ISO端点支持选择
D3	HBTxE	R	当设置位“1”表示高带宽接收ISO端点支持选择
D2	DynFIFO Sizing	R	当设置位“1”表示选择动态FIFO大小调整选项
D1	SoftConE	R	总是“1”，表示软连接/断开
D0	UTMI DataWidth	R	表示选择UTML+数据宽度。始终为0，表示8位

NAKLIMIT0

注：HOST模式下

NAKLIMIT0是一个5位寄存器，用于设置帧个数，在端点0超时接收NAK响应流的数量之后。（可以通过TxInterval进行其他终端等效设置和RxInterval注册）

选择的帧的数目是2（m-1）。（其中m是在寄存器中设置的值，有效值为2-16），如果主机收到NAK响应目标比寄存器设置的限制数目多时，端点将暂停。

注：0或1的值表示禁止NAK超时功能

D4	…	D0
	Endpoint 0 NAK Limit(m)	(LSB)
rw	…	rw

TXMAXP

偏移地址：0x100+16*n　　　　　　　　　　复位值：0x0000

TxMax寄存器定义可以通过所选择的TX端点在一个单一操作下被传输的最大数据量。每个TX端点（端点0除外）都有一个TxMaxP寄存器

D12/15		D11	D10	…	D0
m-1			(MSB)	Maximum Payload/transaction	(LSB)
rw	…	rw	rw	…	rw

该寄存器设置的值可以达到1024个字节，但中断和全速操作的同步传输受到USB规范上的数据包大小以及批量的约束

在与包拆分选项相关的端点批量启用的情况下，乘法器m可以达到32并且限定“USB”数据包最大数目（即通过USB传输的数据包），指定有效载荷的单个数据包在传输之前应分开放置在FIFO中。（如果未启用与包拆分选项相关的端点，D15-D13没有实现，D12-D11（如果包括）被忽略）

注：该数据分组被要求是净荷的确切倍数，由位10到位0指明，这是它本身需要为8,16,32,64或（在高速传输的情况下）512个字节。

对于在高速模式下同步/中断运行和与高带宽选项启用相关的端点，m只能是2或3（分别相当于位11集和位12集），并规定此类交易可发生在一个单一微帧的最大数目，如果位11或位12中任何一个非零，则MH1902T将自动分割将写入FIFO的任何数据分组成达2或3的USB包，每包含指定有效载荷（或更小）。最大有效负载对每个事务都是1024个字节，因此允许多达3072个字节中的每帧帧进行传输。（在全速模式或者同步传输高带宽未启用时，位11和位12被忽略）。

写入该寄存器的值表示数据的总量（指定的有效负荷*M），该值不得超过TX端点设置的FIFO大小，且需要双缓冲时不应超过FIFO的一半大小。

TXCSRL

偏移地址：0x102+16*n　　　　　　　　　　复位值：0x00

TXCSRL是一个8位寄存器，用于为通过当前选择的TX端点传输提供控制和状态位。每个配置的TX端点（不包括端点0）都有一个TXCSRL寄存器

注：该寄存器的解释取决于MH1902T是否充当外设或作为主机，用户也应该注意当读取寄存器反映状态作为写入寄存器的结果时的返回值。

在外设模式下：

D7	D6	D5	D4	D3	D2	D1	D0
IncompTx	ClrDataTog	SentStall	SentStall	FlushFIOFO (self-clearing)	UnderRun	FIFO NotEmpty	TxPktRdy
rc	w	rc	rw	w	rc	rc	rw

Bit	Name	W/R	Description
D7	IncompTx	C/R	当端点正在被用于高带宽时，该位被设置来指示大的数据包已经被分成2或3个封包但没有接收到发送所有部件的指令。**注：在任何非同步传输时，此位将总是返回0**
D6	ClrDataTog	W	CPU写1到该位触发端点数据重置为0
D5	SentStall	C/R	当STALL握手发送时设置此位。FIFO被刷新并且TxPktRdy位清零（见下文），CPU清除此位
D4	SentStall	W/R	CPU写1到该位发出STALL握手指令，CPU清除此位来终止失速状态。**注：该位没有任何端点被用于同步传输**

D3	FlushFIOFO	W	CPU写1到该位来刷新端点FIFO最新的数据包。FIFO指针复位，TxPktRdy位（及以后）被清除并且产生一个中断，同时可以设置TxPktRdy来中止当前正在加载到FIFO的数据包。**注：当TxPktRdy设置时FlushFIFO才会被用到，其他时候可能会导致数据被破坏。还要注意的是如果FIFO双缓冲，FlushFIFO可能需要设置两次来完全清除FIFO**
D2	UnderRun	C/R	在接收到TxPktRdy没有设置的指令时USB设置此位。CPU应该清除此位
D1	FIFO NotEmpty	C/R	在TX FIFO中至少一个包时USB设置此位
D0	TxPktRdy	W/R	CPU加载数据包到FIFO中后，设置此位。当一个数据包发送后它会自动清零，此时产生中断（如果使能）。TxPktRdy也会自动清除之前加载到FIFO来进行双缓冲的第二个包

在主机模式下：

D7	D6
NAK Timeout/ IncompTx	ClrDataTog
rc	w

D5	D4	D3	D2	D1	D0
SentStall	SentStall	FlushFIOFO (self-clearing)	UnderRun	FIFO NotEmpty	TxPktRdy
rc	rw	w	rc	rc	rw

Bit	Name	W/R	Description
D7	NAK Timeout/ IncompTx	C/R	只有批量端口：在TX终点收到NAK响应时间比设定的TxInterval寄存器的NAK限制的时间长时该位被设置。CPU清除此位以允许端点继续。 只有高带宽中断端点：该位将被设置如果数据包被发送到接收设备无响应时
D6	ClrDataTog	W	CPU写1到该位触发端点数据重置为0
D5	SentStall	C/R	当接收到STALL握手时设置此位。此位被设置，任何正在进行中的DMA请求被停止时，FIFO完全刷新并且TxPktRdy位清零（见下文），CPU清除此位
D4	SentStall	W/R	在TxPktRdy位被设置，发送一个SETUP指令来代替OUT指令的同时，CPU设置该位。 **注：设置该位也会清除数据切换**
D3	FlushFIOFO	W	CPU写1到该位来刷新端点FIFO最新的数据包。FIFO指针复位，TxPktRdy位（及以后）被清除并且产生一个中断，同时可以设置TxPktRdy来中止当前正在加载到FIFO的数据包。**注：当TxPktRdy设置时FlushFIFO才会被用到，其他时候可能会导致数据被破坏。还要注意的是如果FIFO双缓冲，FlushFIFO可能需要设置两次来完全清除FIFO**
D2	UnderRun	R/C	已经3次尝试发送一个数据包和没有握手报文被接收时USB设置此位。此位被设置时产生一个中断，TxPktRdy被清除，FIFO完全刷新。CPU应清除此位仅当端点在批量或中断模式下工作时
D1	FIFO NotEmpty	R/C	在TX FIFO中至少一个包时USB设置此位
D0	TxPktRdy	W/R	CPU加载数据包到FIFO中后，设置此位。当一个数据包发送后它会自动清零，此时产生中断（如果使能）。TxPktRdy也会自动清除之前加载到FIFO来进行双缓冲的第二个包

TXCSRH

偏移地址：0x103+16*n 复位值：0x00

TXCSRH是一个8位寄存器，可提供额外的控制，用于通过当前所选的TX端点传输。每个TX端点（不包括端点0）都配置有一个TXCSRH寄存器。

注：该寄存器的解释取决于MH1902T是否充当外设或主机。用户也应该注意当读取寄存器反映状态作为写入寄存器的结果时的返回值。

在外设模式下：

D7	D6	D5	D4	D3	D2	D1	D0
AutoSet	ISO	Mode	DMAReqEnab	FrcDataTog	DMAReqMode	-	-
rw	rw	rw	rw	rw	rw	r	r

Bit	Name	W/R	Description
D7	AutoSet	W/R	如果CPU设置此位，TxPktRdy会在最大数据包被加载到TX FIFO时自动设置。如果装载的分组小于最大分组，TxPktRdy需要手动设置 **注：不为任何高带宽同步端点和高带宽中断端点设置**
D6	ISO	W/R	CPU设置此位来启动TX端点同步传输，并清除TX端点的批量启用或中断转让。 **注：该位只在外设模式下起作用，在主机模式下，总是返回0**
D5	Mode	W/R	CPU设置该位使端点指向TX，清除该位使端点指向RX **注：该位只有在相同的端点FIFO用于TX和RX交易时起作用**
D4	DMAReqEnab	W/R	CPU设置此位来通过DMA的请求启用TX端点
D3	FrcDataTog	W/R	CPU设置该位来强制端点数据触发切换，不管是否接收到ACK，在FIFO的数据分组都被清零。这可以通过同步端点通信速率反馈来中断TX端点。
D2	DMAReqMode	R	CPU设置该位选择DMA请求模式1和清除它来选择DMA请求模式0 **注：在DMAReqEnab被清除的同时和之前，该位不能被清除**
D1-D0	-	R	未使用，始终返回0

在主机模式下：

D7	D6	D5	D4	D3	D2	D1	D0
AutoSet	-	Mode	DMAReqEnab	FrcDataTog	DMAReqMode	Data Toggle Wr.Enable (self-clearing)	Data Toggle
rw	r	rw	rw	rw	rw	w	rw

Bit	Name	W/R	Description
D7	AutoSet	W/R	如果CPU设置此位，TxPktRdy会在最大数据包被加载到TX FIFO时自动设置。如果装载的分组小于最大分组，TxPktRdy需要手动设置 **注：不为任何高带宽同步端点和高带宽中断端点设置**
D6	-	R	未使用，总是返回0
D5	Mode	W/R	CPU设置该位使端点指向TX，清除该位使端点指向RX **注：该位只有在相同的端点FIFO用于TX和RX交易时起作用**
D4	DMAReqEnab	W/R	CPU设置此位来通过DMA的请求启用TX端点
D3	FrcDataTog	W/R	CPU设置该位来强制端点数据触发切换，不管是否接收到ACK，在FIFO的数据分组都被清零。这可以通过同步端点通信速率反馈来中断TX端点。
D2	DMAReqMode	W/R	CPU设置该位选择DMA请求模式1和清除它来选择DMA请求模式0

			注：在DMAReqEnab被清除的同时和之前，该位不能被清除
D1	Data Toggle Write Enable	W	CPU写1到该位，使TX端点数据触发写入（见数据触发位，下同），一旦新的值写入该位会自动清零。
D0	Data Toggle	W/R	读取时，该位表示TX端点数据触发的当前状态。如果D1高，该位随着数据触发所需的设置可能被写入。如果D1低，写入该位的任何值将被忽略

RXMAXP

偏移地址：0x104+16*n 复位值：0x0000

所述RXMAXP寄存器定义数据，可以通过所选择的接收端点在一个单一的传输的最大量操作。每个接收端点（端点0除外）都有一个RXMAXP寄存器。

D12/15	…	D11	D10	…	D0
m-1			(MSB)	Maximum Payload/transaction	(LSB)
rw	…	rw	rw	…	rw

位10:0定义（字节）在一个事务传送的最大有效载荷。设置的值可以达到1024个字节，但受所放置的USB规范上中断和全速操作的同步传输的数据包大小批量限制。

在与包拆分选项相关的端点批量启用的情况下，乘法器m可以达到32并且限定被合并到FIFO中指定有效载荷的USB单一数据包的最大数目。（如果未启用所述分组拆分选项，D15-D13没有实现，D12-D11（如果包括）被忽略）

从位0写到位10的值（在高带宽同步传输的情况下乘以m）必须与在标准端点描述符合的相关端点wMaxPacketSize字段中给出的数值一致。不匹配的话可能会导致意想不到的结果。

写入该寄存器的值表示数据总量（指定的有效负荷*M），该值不得超过RX端点的FIFO大小，若是双缓冲，则不应超过FIFO大小的一半。

注：RxMaxP必须为产生中断的DMA模式1设置偶数字节

RXCSRL

偏移地址：0x106+16*n 复位值：0x00

RXCSRL是一个8位寄存器，用于为当前通过选定接收端点的传输提供控制和状态位。每个配置的接收端点（不包括端点0）都配置有一个RXCSRL寄存器。

注：该寄存器的解释取决于MH1902T是否充当外设或主机。用户也应该注意当读取寄存器反映状态作为写入寄存器的结果时的返回值。

在外设模式下：

D7	D6	D5
ClrDataTog	SentStall	SendStall
w	rc	rw

D4	D3	D2	D1	D0
FlushFIFO (self-clearing)	DataError	OverRun	FIFOFull (self-clearing)	RxPktRdy
w	r	rc	r	rc

Bit	Name	W/R	Description
D7	ClrDataTog	W	CPU写1到该位触发数据，重置为0
D6	SentStall	C/R	当STALL握手发送时此位被设置，CPU应清除此位
D5	SendStall	W/R	CPU写1到该位发出STALL握手，CPU清除该位来终止失速状态。**注：该位没有任何端点被用于同步传输效果**
D4	FlushFIFO	W	CPU写1到该位，从端点FIFO接收读取下一个缓冲数据包，FIFO被复位，RxPktRdy位（及以下）被清除。 **注:如果FIFO双缓冲，FlushFIFO可能需要设置两次完全清楚FIFO**

D3	DataError	R	当数据包中含有CRCRxPktRdy被设置或发生位填充错误RxPktRdy被清除时设置此位 **注：该位仅在IOS模式下运行，在批量模式下，它总是返回0**
D2	OverRun	C/R	当OUT包不能被加载到RxFIFO该位被设置。CPU应清除此位。 **注：该位仅在IOS模式下运行，在批量模式下，它总是返回0**
D1	FIFOFull	R	没有数据包装入RxFIFO中时该位被设置
D0	RxPktRdy	C/R	接收数据分组时该位被设置，当包在FIFO卸载时CPU清除此位，会产生一个中断

在主机模式下：

D7	D6	D5
ClrdataTog	RxStall	ReqPkt
w	rc	rw

D4	D3	D2	D1	D0
FlushFIFO (self-clearing)	DataError/ NAK Timeout	Error	FIFOFull (self-clearing)	RxPktRdy
w	rc	rc	r	rc

Bit	Name	W/R	Description
D7	ClrdataTog	W	CPU写1到该位触发数据，重置为0
D6	RxStall	C/R	当STALL握手发送时此位被设置，CPU应清除此位
D5	ReqPkt	W/R	CPU写1到该位，要求提供事务。当RxPktRdy设置时该位被清零
D4	FlushFIFO	W	CPU写1到该位，从端点FIFO接收读取下一个缓冲数据包，FIFO被复位，RxPktRdy位（及以下）被清除。 **注:当RxPktRdy设置时FlushFIFO才被使用，在其他时候，它可能会导致数据被破坏。如果FIFO双缓冲，FlushFIFO可能需要设置两次完全清楚FIFO**
D3	DataError/ NAK Timeout	C/R	在ISO模式下工作，当数据包中含有CRCRxPktRdy被设置或发生位填充错误RxPktRdy清除时该位被设置。在批量模式中，当接收到NAK响应比设定的RxInterval寄存器的NAK限制的时间长暂停时该位将被设置。CPU清除该位以允许端点继续。但是如果双数据包缓冲单独启用时不允许继续传输。在这种情况下，reqpkt位也应该在相同的周期设定来清除此位
D2	Error	C/R	当3次尝试接收数据但并未接收到数据包时USB设置此位。CPU清除此位，此位被设置时会产生一个中断。
D1	FIFOFull	R	没有数据包装入RxFIFO中时该位被设置
D0	RxPktRdy	C/R	接收数据分组时该位被设置，当包在FIFO卸载时CPU清除此位，会产生一个中断

RXCSRH

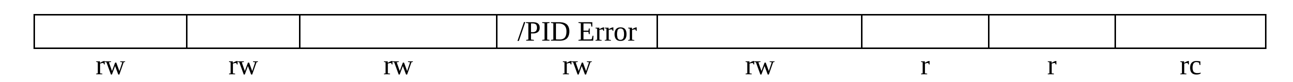
偏移地址：0x107+16*n　　　　复位值：0x00

RXCSRH是一个8位寄存器，通过当前选择的接收终端提供额外的控制和状态位转移。每个接收终端（不包括端点0）都配置有一个RXCSRH寄存器。

注：该寄存器的解释取决于MH1902T是否充当外设或主机。用户也应该注意当读取寄存器反映状态作为写入寄存器的结果时的返回值。

在外设模式下：

D7	D6	D5	D4	D3	D2	D1	D0
AutoClear	ISO	DMAReqEnab	DisNyet	DMAReqMode	-	-	IncompRx



			/PID Error				
rw	rw	rw	rw	rw	r	r	rc

Bit	Name	W/R	Description
D7	AutoClear	W/R	当RxMaxP字节的数据包从RX FIFO端点卸载，RxPktRdy被自动清零时，CPU设置该位。当小于最大分组的分组被卸载时，RxPktRdy需手动清除。当使用一个DMA卸载RXFIFO时，不管RxMaxP，数据从RXFIFO读取四个字节块 **注：不应为高带宽同步端点设置**
D6	ISO	W/R	CPU设置该位来接收端点的同步传输，并将该位清除来接收端点的块或中断传输
D5	DMAReqEnab	W/R	DMA请求RX端点时CPU设置该位
D4	DisNyet /PID Error	W/R	分裂/中断交易：CPU设置该位来禁止发送NYET握手。设置后，所有成功接收包括FIFO满点的数据包被设置为ACK’d **注：该位只有在高速模式下有影响，在这种模式下，它应该对所有中断端点进行设置。** ISO事务：核心设置该位表示接收到的数据包的PID误差
D3	DMAReqMode	W/R	CPU设置该位来选择DMA请求模式1，清除该位来选择DMA请求模式0
D2-D1	-	R	未使用，始终返回0
D0	IncompRx	C/R	该位在一个高带宽同步/中断传送设置，如果由于没有接收部分数据，RxFIFO的包是不完全的。当RxPktRdy清零时该位也被清除 **注：在任何非同步传输时，此位总是返回0**

在主机模式下：

D7	D6	D5	D4	D3	D2	D1	D0
AutoClear	AutoReq	DMAReqEnab	PID Error	DMAReqMode	Data Toggle Wr. Enable (self-clearing)	Data Toggle	IncompRx
rw	rw	rw	r	rw	r	r	rc

Bit	Name	W/R	Description
D7	AutoClear	W/R	当RxMaxP字节的数据包从RX FIFO端点卸载，RxPktRdy被自动清零时，CPU设置该位。当小于最大分组的分组被卸载时，RxPktRdy需手动清除。当使用一个DMA卸载RXFIFO时，不管RxMaxP，数据从RXFIFO读取四个字节块 **注：不应为高带宽同步端点设置**
D6	AutoReq	W/R	RxPktRdy位清零时ReqPkt位自动设置，CPU设置该位。 **注：接收到短报文时，该位自动清零。**
D5	DMAReqEnab	W/R	DMA请求RX端点时CPU设置该位
D4	PID Error	R	ISO事务：核心设置该位表示接收到的数据包的PID误差 分裂/中断交易：该位设置被忽略
D3	DMAReqMode	W/R	CPU设置该位来选择DMA请求模式1，清除该位来选择DMA请求模式0
D2	Data Toggle Wr. Enable	R	CPU写1到该位，触发端点0当前状态数据写入（见数据触发位，下同）。一旦新的值写入，该位会自动清零。
D1	Data Toggle	R	读取时，该位指示端点0的数据触发当前状态。如果D10高，该位可能会被写入数据切换需要的设定，如果D10低，写入该位的任何值将被忽略

D0	IncompRx	C/R	该位在高带宽同步或中断传输中接收到的数据包不完整时设置。当RxPktRdy被清除时该位被清除。 **注：如果遵循USB协议，该位不应该设置。（在任何非同步传输下，该位始终返回0）**

RXCOUNT

偏移地址：0x108+16*n 复位值：0x0000

RXCOUNT是一个14位的只读寄存器，它预留数据字节中当前被Rx FIFO读出的数据包的数量。如果该分组作为多个数据包散包被发送时，给出的数字将用于组合分组。

注：当RxPktRDY(RxCSR.D0)设置时，返回的值更改为FIFO卸载是唯一有效的

D13	…	D0
(MSB)	Endpoint Rx Count	(LSB)
r	…	r

TXTYPE

偏移地址：0x10A+16*n 复位值：0x00

TXTYPE是写入到端点目标端点号的8位寄存器，该事务协议用于当前所选的TX端点，以及它的运行速度。每个配置的TX端点（端点0有自己的寄存器类型，为1Ah）都有一个TxType寄存器

D7	D6	D5	D4	D3	D2	D1	D0
*Speed		Protocol		Target Endpoint Number			
rw	rw	rw	rw	rw	rw	rw	rw

Bit	Name	W/R	Description
D7-D6	Speed	W/R	当芯配置了多点选项的目标设备的操作速度： 00：未使用（注：如果选中，则目标将被认为是与核心使用相同的连接速度） 10：全 11：低 当内核没有配置多点选项时，这些位不能被访问
D5-D4	Protocol	W/R	该CPU设置这个选择为接收终端所需的协议： 00：控制 01：同步 10：散装 11：中断
D3-D0	Target Endpoint Number	W/R	CPU应将该值设置为包含在设备枚举期间返回到MH1902T在RX端点描述的端点号

TXINTERVAL

偏移地址：0x10B+16*n 复位值：0x00

TXINTERVAL是一个8位寄存器，用于中断和同步传输，为当前selectRx端点定义轮询间隔。对于批量端点，该寄存器设置帧之后，端点应当超时收到NAK响应流的数量。

D7	…	D0
Tx Polling Interval/NAK Limit (m)		
rw	…	rw

Transfer Type	Speed	Valid values (m)	Interpretation

Interrupt	低速或全速	1 – 255	轮转间隔为m帧
Isochronous	全速	1 – 16	轮转间隔为2(m-1)帧
Bulk	全速	2 – 16	NAK 限制为 2(m-1)帧注：0或1的值禁用NAK超时FUNC

RXTYPE

偏移地址：0x10C+16*n 复位值：0x00

RXTYPE是写入到端点目标端点号的8位寄存器，该事务协议用于当前所选接收端点，以及它的运行速度。每个配置的接收端点（端点0有自己的寄存器类型）都有一个RxType寄存器

D7	D6	D5	D4	D3	D2	D1	D0
*Speed		Protocol		Target Endpoint Number			
rw	rw	rw	rw	rw	rw	rw	rw

Bit	Name	W/R	Description
D7-D6	Speed	W/R	当芯配置了多点选项的目标设备的操作速度： 00：未使用（注：如果选中，则目标将被认为是与核心使用相同的连接速度） 10：全 11：低 当内核没有配置多点选项时，这些位不能被访问
D5-D4	Protocol	W/R	该CPU设置这个选择为接收终端所需的协议： 00：控制 01：同步 10：散装 11：中断
D3-D0	Target Endpoint Number	W/R	CPU应将该值设置为包含在设备枚举期间返回到MH1902T在RX端点描述的端点号

RXINTERVAL

偏移地址：0x10D+16*n 复位值：0x00

RXINTERVAL是一个8位寄存器，用于中断和同步传输，为当前selectRx端点定义轮询间隔。对于批量端点，该寄存器设置帧之后，端点应当超时收到NAK响应流的数量。每个接收端点（端点0除外）都配置有一个RXINTERVAL寄存器

D7	…	D0
Rx Polling Interval/NAK Limit (m)		
rw	…	rw

Transfer Type	Speed	Valid values (m)	Interpretation
Interrupt	低速或全速	1 – 255	轮转间隔为m帧
Isochronous	全速	1 – 16	轮转间隔为2(m-1)帧
Bulk	全速	2 – 16	NAK 限制为 2(m-1)帧注：0或1的值禁用NAK超时FUNC

FIFOSIZE

偏移地址：0x10F+16*n 复位值：0x00

FIFOSIZE是一个8位只读寄存器，用来返回与所选附加TX/RX端点相关的FIFO的大小。下半字节进行编码所选择的TX端点FIFO的大小，上半字节进行编码所选择的接收端点FIFO的大小。3-13的值对应于2^n个字节的FIFO大小（8-8192字节）。如果端点尚未配置，0值会显示出来，其中TX和RX端点共享相同的FIFO，在RX FIFO的大小将被编码为0xF

注：该寄存器只有这种解释当索引寄存器设置选择端点1-15中的一个和动态调整大小没有被选中时，它具有当索引寄存器设置选择端点0的一个特殊的解释，而返回的结果是无效的，其中动态的FIFO大小被使用。

D7	…	D4	D3	…	D0
Rx FIFO Size			Tx FIFO Size		
r	…	r	r	…	r

19.4.4 FIFOx

偏移地址：0x20-0x5F 复位值：0x00000000

这个地址范围提供了CPU访问每个端点的FIFO的16个地址，写这些地址将数据加载到TXFIFO中相应的端点。从这些地址可以在RXFIFO相应的端点读取或者卸载数据

地址范围是20H-5Fh的和FIFO都位于32位双字边界（端点0-20，端点1-24，端点15在5Ch）

注：（i）往返的FIFO可以根据需要设置8位，16位或32位，和访问的任何组合允许提供的数据访问是连续的。然而，与一个分组相关联的传送必须是相同宽度，使得数据是一致的按字节，字处理或双字对齐。然而最后的传输可能为完成一个奇数字节或奇字传输而传输比以前的传输较少的字节

（ii）根据FIFO和期望的最大数据包大小的尺寸，所述的FIFO支持单包或双包缓冲。但是，不支持写入后需要进行设置的多个分组进行突发写入

（iii）经过STALL响应或TX 三振错误的端点1-15，相关的FIFO完全刷新

19.4.5 多点控制/状态寄存器(Additional Multipoint Control/Status registers)

TXFUNCADDR/RXFUNCADDR

偏移地址：0x80+8*n/0x84+8*n 复位值：0x00/0x00

TXFUNCADDR/RXFUNCADDR是7位读/写寄存器，记录目标函数是通过相关联的端点（EPN）访问的地址。TXFUNCADDR需要对每个用到的TX端点定义，RXFUNCADDR需要对每个使用到的RX接收端点定义

注：TXFUNCADDR必须为端点0定义，RXFUNCADDR寄存器上不存在端点0

D6	…	D0
Address of Target Function		
rw	…	rw

TXHUBADDR/RXHUBADDR

偏移地址：0x82+8*n/0x86+8*n 复位值：0x00/0x00

注：只在主机模式下

TXHUBADDR/RXHUBADDR是8位读/写寄存器，像TxHubPort和RxHubPort，只需要被写入通过USB2.0集线器的全速设备并连接到TX/RX端点EPN，进行必要事务在全速/低速传输之间进行转换。

D7	D6	…	D0
Multiple Translators	Hub Address		
rw	rw	…	rw

TXHUBPORT/RXHUBPORT

偏移地址：0x83+8*n/0x87+8*n 复位值：0x00/0x00

TXHUBPORT和RXHUBPORT只需要写入其中全速或低速设备通过高速USB2.0集线器携带必要的事务转换连接到TX/RX端点的EPN。这样的情况下，这些7位读/写寄存器需要被用于记录通过与该端点相关联的目标被访问的USB2.0集线器端口

D6	…	D0
Hub Port		
rw	…	rw

19.4.6 控制/状态寄存器(Additional Control/Status registers)

VCONTROL

偏移地址：0x68 复位值：-

VCONTROL是一个可选的被包含在被配置芯片的UTMI+ PHY的供应商寄存器，其大小也可配置，高达32位。寄存器的结构是由系统设计者设计。尽管用户应注意的是VCONTROL寄存器由UTML+规范定义

VSTATUS

偏移地址：0x68 复位值：-

VSTATUS是一个可选的被包含在被配置芯片的UTMI+ PHY的供应商寄存器，其大小也可配置，高达32位。寄存器的结构是由系统设计者设计。尽管用户应注意的是VSTATUS寄存器由UTML+规范定义，但该寄存器可以在地址68H进行访问

用户还应该注意的是：

（1） VSTATUS每6个XCLK周期输入一次总线采样一次

（2） PHY改变VSTATUS输入总线和新值之间的延迟从VSTATUS寄存器中读取2Hc+ Xc到3HC+6Xc之间。其中Hc是XCLK的一个循环周期

HWVERS

偏移地址：0x6C 复位值：Version dependent

HWVERS 寄存器是一个 16 位的只读寄存器，从中产生并返回有关核心硬件信息，特别是 RTL 版本号（vxx.yyy）和版本信息

D15	D14	…	D10	D9	…	D0
RC	xx			yyy		
r	r	…	r	r	…	r

Bit	Name	W/R	Description
D15	RC	R	设置为“1”，如果是发行候选版本核心RTL而不是一个完整版本的核心
D14-D10	xx	R	主版本号（范围0-31）
D9-D0	yyy	R	副版本号（范围0-999）

19.4.7 配置寄存器(Configuration registers)

EPINFO

偏移地址：0x78 复位值：Implementation dependent

该8位只读寄存器允许包括在设计的回读的TX和RX端点的数量

D7	D6	D5	D4	D3	D2	D1	D0
RxEndPoints				TEndPoints			

r	r	r	r	r	r	r	r

Bit	Name	W/R	Description
D7-D4	RxEndPoints	R	接收的数目，在设计的端点来实现
D3-D0	TEndPoints	R	TX的数目，在设计的端点来实现

RAMINFO

偏移地址：0x79 复位值：Implementation dependent
该8位只读寄存器提供对RAM的宽度信息

D7	D6	D5	D4	D3	D2	D1	D0
DMAChans				RamBits			
r	r	r	r	r	r	r	r

Bit	Name	W/R	Description
D7-D4	DMAChans	R	在设计实施的DMA通道的数目
D3-D0	RamBits	R	RAM地址总线的宽度

LINKINFO

偏移地址：0x7A 复位值：0x5C
这8位寄存器允许指定一些延误

D7	D6	D5	D4	D3	D2	D1	D0
WTCON				WTID			
rw	rw	rw	rw	rw	rw	rw	rw

Bit	Name	W/R	Description
D7-D4	WTCON	W/R	设置要应用的等待，以待在533.3ns单位用户的连接/断开滤波器（默认设置对应于2.667us）
D3-D0	WTID	W/R	设置延迟从IDPULLUP应用被断言IDDIG被认为是有效的4.369ms的单位（默认设置对应于52.43ms）

VPLEN

偏移地址：0x7B 复位值：0x3C
这个8位寄存器设置的VBUS脉冲充电的持续时间

D7	D6	D5	D4	D3	D2	D1	D0
VPLEN							
rw	rw	rw	rw	rw	rw	rw	rw

Bit	Name	W/R	Description
D7-D0	VPLEN	W/R	设置在546.1微秒位单位的VBUS脉冲充电的持续时间（默认设置对应为32.77ms）

FS_EOF1

偏移地址：0x7D 复位值：0x77
这个8位寄存器设置的最小时间间隔在最后一个事务开始到EOF全速事务之间即可

D7	D6	D5	D4	D3	D2	D1	D0
(msb)			FS_EOF1				(lsb)
rw	rw	rw	rw	rw	rw	rw	rw

Bit	Name	W/R	Description
D7-D0	FS_EOF1	W/R	设置了全速事务停止EOF开始新事务的时间以533.3ns为单位（默认设置对应于63.46us）

LS_EOF1

偏移地址：0x7E 复位值：0x72

这8位寄存器设置最小时间间隔在最后一个事务开始到EOF低速事务之间即可

D7	D6	D5	D4	D3	D2	D1	D0
(msb)			LS_EOF1				(lsb)
rw	rw	rw	rw	rw	rw	rw	rw

Bit	Name	W/R	Description
D7-D0	LS_EOF1	W/R	设置了低转速事务停止EOF开始新事务的时间以1.067ns为单位（默认设置对应于121.6us）

SOFT_RST

偏移地址：0x7F 复位值：0x00

这个8位寄存器将置低输出复位信号NRSTO和NRSTOX。该寄存器会自我清除，并输入NRST复位

Unused	D1	D0
(msb) SOFT_RST		(lsb)
	rw	rw

Bit	Name	W/R	Description
D7-D2	-	-	预留，总是返回0
D1	NRSTX	W/R	该位的默认值是1’b0;当1写入该位时，输出NRSTXO将被置（low）内的CLK输入的7个周期的最小延迟。NRSTXO将被异步断言和同步输出，相对于XCLK被拉高。该寄存器是自我清除，并输入NRST复位
D0	NRST	W/R	该位的默认值是1’b0;当1写入该位时，输出NRSTXO将被置（low）内的CLK输入的7个周期的最小延迟。NRSTXO将被异步断言和同步输出，相对于XCLK被拉高。该寄存器是自我清除，并输入NRST复位

19.4.8 扩展寄存器(Extended registers)

RQPKTCOUNT

偏移地址：0x300+2*n 复位值：0x0000

对于每一个接收端点1-15中，MH1902T提供一个16位寄存器RQPKTCOUNT。该读/写寄存器用于在主机模式下，来指定在长度MAXP的一个或多个散装分组的块传输到RX端点数据包N待转移的数目。芯使用记录在该寄存器中的值来确定请求的发出，其中AutoReq选项（包括在RxCSR寄存器）已设置数量。

D15	…	D0

(msb)	RqPktCount	(lsb)
rw	…	rw

Bit	Name	W/R	Description
D15-D0	RqPktCount	W/R	设置块传送的包的数量的MAXP大小，当AutoReq未设置时只在主机模式下使用

双缓存包失能

19.4.8.1.1 RX DPKTBUFDIS

偏移地址：0x340 复位值：0x0000

RX DPKTBUFDIS是一个16位寄存器，表示接收的哪些端点已禁用MH1902T产品规范的双包缓冲功能

注：没有被配置的端点位可以断言写一个“1”到各自的寄存器，但是禁用位不会有任何显著的影响

D15	D14	D13	D12	D11	D10	D9	D8
EP15 RxDis	EP14 RxDis	EP13 RxDis	EP12 RxDis	EP11 RxDis	EP10 RxDis	EP9 RxDis	EP8 RxDis
rw	rw	rw	rw	rw	rw	rw	rw

D7	D6	D5	D4	D3	D2	D1	D0
EP7 RxDis	EP6 RxDis	EP5 RxDis	EP4 RxDis	EP3 RxDis	EP2 RxDis	EP1 RxDis	EP0 RxDis
rw	rw	rw	rw	rw	rw	rw	r

Bit	Name	W/R	Description
D15	EP15 RxDis	W/R	Rx端点15中断
D14	EP14 RxDis	W/R	Rx端点14中断
D13	EP13 RxDis	W/R	Rx端点13中断
D12	EP12 RxDis	W/R	Rx端点12中断
D11	EP11 RxDis	W/R	Rx端点11中断
D10	EP10 RxDis	W/R	Rx端点10中断
D9	EP9 RxDis	W/R	Rx端点9中断
D8	EP8 RxDis	W/R	Rx端点8中断
D7	EP7 RxDis	W/R	Rx端点7中断
D6	EP6 RxDis	W/R	Rx端点6中断
D5	EP5 RxDis	W/R	Rx端点5中断
D4	EP4 RxDis	W/R	Rx端点4中断
D3	EP3 RxDis	W/R	Rx端点3中断
D2	EP2 RxDis	W/R	Rx端点2中断
D1	EP1 RxDis	W/R	Rx端点1中断
D0	Unused	R	预留

19.4.8.1.2 TX DPKTBUFDIS

偏移地址：0x342 复位值：0x0000

TX DPKTBUFDIS是一个16位寄存器，表示TX哪个端点已禁用MH1902T产品规范的双包缓冲功能

注：没有被配置的端点位可以断言写一个“1”到各自的寄存器，但是禁用位不会有任何显著的影响

D15	D14	D13	D12	D11	D10	D9	D8
EP15	EP14	EP13	EP12	EP11	EP10	EP9	EP8

TxDis	TxDis	TxDis	TxDis	TxDis	TxDis	TxDis	TxDis
rw	rw	rw	rw	rw	rw	rw	rw

D7	D6	D5	D4	D3	D2	D1	D0
EP7 TxDis	EP6 TxDis	EP5 TxDis	EP4 TxDis	EP3 TxDis	EP2 TxDis	EP1 TxDis	EP0 TxDis
rw	rw	rw	rw	rw	rw	rw	r

Bit	Name	W/R	Description
D15	EP15 TxDis	W/R	Tx端点15中断
D14	EP14 TxDis	W/R	Tx端点14中断
D13	EP13 TxDis	W/R	Tx端点13中断
D12	EP12 TxDis	W/R	Tx端点12中断
D11	EP11 TxDis	W/R	Tx端点11中断
D10	EP10 TxDis	W/R	Tx端点10中断
D9	EP9 TxDis	W/R	Tx端点9中断
D8	EP8 TxDis	W/R	Tx端点8中断
D7	EP7 TxDis	W/R	Tx端点7中断
D6	EP6 TxDis	W/R	Tx端点6中断
D5	EP5 TxDis	W/R	Tx端点5中断
D4	EP4 TxDis	W/R	Tx端点4中断
D3	EP3 TxDis	W/R	Tx端点3中断
D2	EP2 TxDis	W/R	Tx端点2中断
D1	EP1 TxDis	W/R	Tx端点1中断
D0	EP0 TxDis	R	预留

C_T_UCH

偏移地址：0x344　　　　　　　　　　　复位值：Various

该寄存器设置在高速恢复信号（作为主机）结束的延迟时使UTM恢复正常工作模式。这个号码乘以4表示在超时发生前XCLK的周期数。也就是说，如果XCLK为30MHz，此数字表示在超时发生前的33.3ns的时间间隔数目。如果XCLK为60MHz，这个数字代表在超时之前的16.7ns的时间间隔数。虽然该位在CLK域由主机写入，在XCLK领域利用这个值的计数器，没有时间域交叉设置，在此寄存器中的值是静态的，默认值是位于配置文件musbhsfc_xcfg.v名称相同的编译器指令的值。

D15	…	D0
	C_T_UCH[15:8]	
rw	…	rw

Bit	Name	W/R	Description
D15-D0	C_T_UCH	W/R	从高速的末端的延迟恢复信令恢复UTM正常工作模式。默认值是由musbhsfc_xcfg.v文件编译器指令来确定。如果主机PHY数据宽度为16位时默认值是2F3h（XCLK为30MHz），PHY数据宽度时8位对应100us延迟时默认值是5E6H

19.4.9 DMA 寄存器(DMA registers)

DMA_INTR

偏移地址：0x200　　　　　　　　　　　复位值：0x00

该寄存器提供了中断每个DMA通道。阅读时，该中断寄存器清零。当该寄存器的任意位被设置时，输出DMA_NINT被置为低电平，导致发生中断事件，在第17（可选的DMA控制器的说明）进行说明。该寄存器将被设置在DMA中断使能位对应通道启用。

D7	D6	D5	D4	D3	D2	D1	D0
DMA_INTR[7:0]							
rw	rw	rw	rw	rw	r	r	r

Bit	Name	W/R	Description
D7	CH8 DMA_INTR	W/R	通道8DMA中断
D6	CH7 DMA_INTR	W/R	通道7DMA中断
D5	CH6 DMA_INTR	W/R	通道6DMA中断
D4	CH5 DMA_INTR	W/R	通道5DMA中断
D3	CH4 DMA_INTR	W/R	通道4DMA中断
D2	CH3 DMA_INTR	R	通道3DMA中断
D1	CH2 DMA_INTR	R	通道2DMA中断
D0	CH1 DMA_INTR	R	通道1DMA中断

DMA_CNTL

偏移地址：0x204 复位值：0x0000

该寄存器仅在MH1902T被配置使用至少一个内部DMA通道时使用。该寄存器为每个通道提供了DMA传输控制。启用，传输方向，传输模式和DMA突发模式都由该寄存器控制。

D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
DMA_BRSTM		DMA_ERR	DMAEP				DMAIE	DMAMODE	DMA_DIR	DMA_EN
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit	Name	W/R	Description
D10-D9	DMA_BRSTM	W/R	突发模式： 00=突发模式0：未指定长度的突发 01=突发模式1：INCR4或未指定长度 10=突发模式2：INCR8,INCR4或未指定长度 11=突发模式3：INCR16,INCR8,INCR4或未指定长度
D8	DMA_ERR	W/R	总线错误位：表示总线错误已被观察的输入AHB_HRESPM[1:0].该位由软件清零
D7-D4	DMAEP	W/R	此信道分配给该端点号
D3	DMAIE	W/R	DMA中断使能
D2	DMAMODE	W/R	该位选择DMA传输模式： 0=DMA模式0传输 1=DMA模式1传输
D1	DMA_DIR	W/R	该位选择DMA传输方向： 0=DMA写（RX端点） 1=DMA读（TX端点）
D0	DMA_EN	W/R	此位将启动DMA传输，并会导致传输开始

DMA_ADDR

偏移地址：0x208 复位值：0x00000000

该寄存器标识相应的DAM通道的当前存储器地址。写入该寄存器的初始内存地址必须具有一个值，使它的模4的值等于0。即：DMA_ADDR[1:0]必须等于2’b00。该寄存器的低两位是只读的，不能由

软件进行设置。作为DMA传送的进行，存储地址作为字节递增传送。

D31	D30	…	D1	D0
DMA_ADDR[31:0]				
rw	rw	…	rw	rw

Bit	Name	W/R	Description
D31-D0	DMA_ADDR	W/R	DMA的内存地址，请注意：写入该寄存器的初始内存地址必须具有一个值，使它的模4的值等于0。即：DMA_ADDR[1:0]必须等于2'b00。该寄存器的低两位是只读的，不能由软件进行设置

DMA_COUNT

偏移地址：0x20C 复位值：0x00000000

该寄存器用来识别传输的当前DMA计数，软件设置标识整个传输长度的传输的初始计数。随着计数的进展，计数减少以字节为单位转移。

D31	D30	…	D1	D0
DMA_ADDR[31:0]				
rw	rw	…	rw	rw

Bit	Name	W/R	Description
D31-D0	DMA_ADDR	W/R	DMA存储器地址对应的DMA通道。注：如果DMA在计数为0启用时，总线不会被要求，DMA中断产生

19.4.10动态FIFO寄存器(Dynamic FIFO registers)

动态FIFO寄存器仅当MH1902T被配置未使用动态的FIFO大小时可用。有一组寄存器每个端点都不包括0端点。访问这些变址的索引寄存器地址0EH必须设置相应的终点。

TXFIFOSZ

偏移地址：0x62 复位值：0x00

TXFIFOSZ是一个五位寄存器，它控制所选择的TX端点FIFO的大小。

D4	D3	D2	D1	D0
DPB	SZ3	SZ2	SZ1	SZ0
rw	rw	rw	rw	rw

Bit	Name	W/R	Description
D4	DPB	W/R	定义双缓冲包是否支持：当'1'时，支持双包缓冲，当'0'时，仅支持单包缓冲
D3-D0	SZ[3:0]	W/R	被允许的最大数据包大小（散装FIFO内的任何分裂前/高带宽的数据包在传输之前表格）

SZ[3:0]				数据包大小
0	0	0	0	8
0	0	0	1	16
0	0	1	0	32
0	0	1	1	64
0	1	0	0	128

	1	0	1	256
0	1	1	0	512
0	1	1	1	1024
1	0	0	0	2048
1	0	0	1	4096

如果DPB=0，FIFO大小等于这个值，如果DPB=1,FIFO将是该值的两倍

RXFIFOSZ

偏移地址：0x63 复位值：0x00

RXFIFOSZ是一个五位寄存器，它控制所选择的接收端点FIFO的大小。

D4	D3	D2	D1	D0
DPB	SZ3	SZ2	SZ1	SZ0
rw	rw	rw	Rw	rw

Bit	Name	W/R	Description
D4	DPB	W/R	定义双缓冲包是否支持：当'1'时，支持双包缓冲，当'0'时，仅支持单包缓冲
D3-D0	SZ[3:0]	W/R	被允许的最大数据包大小（以下接收批量/高带宽数据包的FIFO中的任意组合之后表格）

SZ[3:0]				数据包大小
0	0	0	0	8
0	0	0	1	16
0	0	1	0	32
0	0	1	1	64
0	1	0	0	128
	1	0	1	256
0	1	1	0	512
0	1	1	1	1024
1	0	0	0	2048
1	0	0	1	4096

如果DPB=0，FIFO大小等于这个值，如果DPB=1,FIFO将是该值的两倍

TXFIFOADD

偏移地址：0x64 复位值：0x0000

TXFIFOADD是一个14位寄存器，它控制所选择的Tx端点FIFO的起始地址

D13	D12	…	D0
-	AD12	…	AD0
rw	rw	rw	rw

Bit	Name	W/R	Description
D13	-	W/R	预留，将来使用
D12-D0	AD[12:0]	W/R	以8字节为单位启动端点的FIFO地址如下：

AD[12:0]				起始地址
0	0	0	0	0000
0	0	0	1	0008
0	0	0	2	0010
		…		…
1	F	F	F	FFF8

RXFIFOADD

偏移地址：0x66 复位值：0x0000
RXFIFOADD是一个14位寄存器，它控制所选择的RX端点FIFO的起始地址

D13	D12	…	D0
-	AD12	…	AD0
rw	rw	rw	rw

<table>
<tr><th>Bit</th><th>Name</th><th>W/R</th><th>Description</th></tr>
<tr><td>D13</td><td>-</td><td>W/R</td><td>预留，将来使用</td></tr>
<tr><td>D12-D0</td><td>AD[12:0]</td><td>W/R</td><td>以8字节为单位启动端点的FIFO地址如下：
<table>
<tr><th colspan="4">AD[12:0]</th><th>起始地址</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0000</td></tr>
<tr><td>0</td><td>0</td><td>0</td><td>1</td><td>0008</td></tr>
<tr><td>0</td><td>0</td><td>0</td><td>2</td><td>0010</td></tr>
<tr><td colspan="4">…</td><td>…</td></tr>
<tr><td>1</td><td>F</td><td>F</td><td>F</td><td>FFF8</td></tr>
</table></td></tr>
</table>

19.4.11LPM 寄存器(LPM registers)

LPM_ATTR

偏移地址：0x360 复位值：0x00
该寄存器用来定义LPM事务和睡眠周期的属性。在主机模式和外设模式下，该寄存器的意义是一样的，不过用于主机和外设时的数据源不同如下：

1) 在外设模式下，该寄存器中的值将包含收到的最后LPM事务，等效属性被接受。如果响应于仅次于LPM事务的ACK，则该寄存器与LPM报的内容更新。
   该寄存器可以通过软件更新，在其他情况下，该寄存器将保持当前值。
2) 在主机模式下，软件将建立在此寄存器中的值来定义将要发送的LPM事务。这些值将被插入到下一LPM事务的有效载荷。

<table>
<tr><td>D15</td><td>D14</td><td>D13</td><td>D12</td><td>D11</td><td>D10</td><td>D9</td><td>D8</td></tr>
<tr><td colspan="4">EndPoint</td><td colspan="3">Reserved</td><td>RmtWak</td></tr>
<tr><td>r</td><td>r</td><td>r</td><td>r</td><td>r</td><td>r</td><td>r</td><td>r</td></tr>
</table>

<table>
<tr><td>D7</td><td>D6</td><td>D5</td><td>D4</td><td>D3</td><td>D2</td><td>D1</td><td>D0</td></tr>
<tr><td colspan="4">HIRD</td><td colspan="4">LinkState</td></tr>
<tr><td>r</td><td>r</td><td>r</td><td>r</td><td>r</td><td>r</td><td>r</td><td>r</td></tr>
</table>

Bit	Name	W/R	Description
D15-D12	EndPnt	R	这是LPM事务令牌包的端点
D11-D9	Reserver	R	预留，读取时总是返回0
D8	RmtWak	R	该位是远程唤醒使能位 RmtWak = 1’b0 :远程唤醒未使用； RmtWak = 1’b1:远程唤醒功能。 该位临时性被设置，并且只应用于当前暂停状态。暂停周期之后，枚举在协商的远程唤醒功能将适用
D7-D4	HIRD	R	这是主机发起恢复持续时间。该值是主机恢复的最短时间，该寄存器中的值对应的时实际恢复的时间。 恢复时间=50微秒+HIRD*75微秒。结果在50-1200微秒之间
D3-D0	LinkState	R	此值由主机或外设用来指示LPM事务收取验证过度的状态。 链路状态=4’h0-预留 链路状态=4’h1-睡眠状态（L1）

			链路状态=4’h2-预留 链路状态=4’h3-预留

LPM_CNTRL

偏移地址：0x362 复位值：0x00

在外设模式下：

D7	D6	D5	D4	D3	D2	D1	D0
			LPMNAK	LPMEN		LPMRES	LPMXMT
r	r	r	rw	rw		rw	rw

Bit	Name	W/R	Description
D7-D5	Reserved	R	预留，读取时总是返回0x00
D4	LPMNAK	W/R	此位是用来存放所有端点的状态来应对此外的所有交易，那么LPM交易将会变成一个NAK。该位会在MH1902T暂停LPM之后起作用，这种情况下，MH1902T将会继续NAK直到这个位在软件中被清除
D3-D2	LPMEN	W/R	该寄存器用于在MH1902T中启用LPM。有三个层次的LPM可以启用，这将决定MH1902T对LPM事务的响应。这三个层次分别是： ● 2’b00，2’b10-不支持LPM扩展事务。在这种情况下，MH1902T将不响应LPM事务并且事务会超时 ● 2’b01-不支持但是会延长LPM事务。在这种情况下，MH1902T将以STALL形式响应LPM事务 ● 2’b11-MH1902T支持LPM扩展事务。在这种情况下，MH1902T将响应一个NYET或由LPMXMT和其他条件的值来确定的ACK
D1	LPMRES	W/R	该位被软件用来启动恢复（远程唤醒）。该位的区别在于，电力寄存器（地址0x01.2）的经典RESUME位，所述RESUME信号定时由硬件控制。在软件中写此位，恢复信号将被置为50微秒，此位自动清除
D0	LPMXMT	W/R	该位被软件设置来指示MH1902T接收到下一个LPM事务时转换到L1的状态。如果LPMEN设置为2’b11该位才有效。该位可以在相同周期的LPMEN进行设置。如果该位设置为1’b1和LPMEN=2’b11,在MH1902T可以通过以下方式应对： ● 如果没有数据正在等待（所有TX FIFO都是空的），该MH1902T会回应一个ACK。在这种情况下，此位将自动清除并且软件会产生中断 ● 如果数据正在等待（至少一个数据驻留在TX FIFO），该MH1902T将响应一个NYET。在这种情况下，该位不会自动清除，但是软件将会产生中断

在主机模式下：

D7	D6	D5	D4	D3	D2	D1	D0
Reserved						LPMRES	LPMXMT
r	r	r	r	r	r	rw	rw

Bit	Name	W/R	Description
D7-D2	Reserved	R	预留，读取时总是返回0
D1	LPMRES	W/R	该位被软件用来启动从L1状态的恢复。该位不同于在功率

			寄存器（地址0x01.2）的经典RESUME位与恢复信号定时由硬件控制。在软件中写此位，恢复信号将被置为HIRD字段中的LPM_ATTR寄存器所指定的时间。该位会自动清除
D0	LPMXMT	W/R	软件设置此位为发送LPM事务，此位会自动清除，会在收到任何指令牌或发生三个超时时立即被清零

LPM_INTREN

偏移地址：0x363 复位值：0x00

LPM_INTREN是一个6位寄存器，用于为LPM_INTR寄存器中的中断提供使能位。如果在这个寄存器中的一个位设置位1，LPM_INTR寄存器中的相应中断被设置时MC_NINT将确认。如果在这个寄存器中的一个位设定为0，LPM_INTR寄存器中的相应中断仍被设置但MC_NINT不会被确认（低电平）。复位时，该寄存器的所有位被重置为0

D7	D6	D5	D4	D3	D2	D1	D0
RESERVED		LPMERREN	LPMRESEN	LPMACKEN	LPMNYEN	LPMSTEN	LPMTOEN
r	r	r	r	rw	rw	rw	rw

Bit	Name	W/R	Description
D7-D6	Reserved	R	预留，读取时总是返回0
D5	LPMERREN	R	1’b0:禁用LPMERR中断 1’b1:启用LPMERR中断
D4	LPMRESEN	R	1’b0:禁用LPMERR中断 1’b1:启用LPMERR中断
D3	LPMACKEN	W/R	1’b0:禁用LPMERR中断 1’b1:启用LPMERR中断
D2	LPMNYEN	W/R	1’b0:禁用LPMERR中断 1’b1:启用LPMERR中断
D1	LPMSTEN	W/R	1’b0:禁用LPMERR中断 1’b1:启用LPMERR中断
D0	LPMTOEN	W/R	1’b0:禁用LPMERR中断 1’b1:启用LPMERR中断

LPM_INTR

偏移地址：0x364 复位值：0x00

LPM_INTR是一个7位寄存器，提供LPM电源状态。当一个位设置为1，输出MC_NINT被占用（低）相应的使能位也被设置为1。如果相应的启用位被设置为0，则输出MC_NINT不被肯定。复位时，寄存器中的所有位复位为0。该寄存器读取时清除。

在外设模式下：

D7	D6	D5	D4	D3	D2	D1	D0
RESERVED		LPMERR	LPMRES	LPMNC	LPMACK	LPMNY	LPMST
r	r	r	r	r	r	r	R

Bit	Name	W/R	Description
D7-D6	Reserved	R	预留，读取时总是返回0X0
D5	LKPMERR	R	如果接收到有一个链路状态不被支持的LPM事务时该位被设

			置。在这种情况下，响应该事务将交给STALL，然而LPM_ATTR寄存器将被更新使软件可以观察到不兼容的LPM数据包的有效负载。
D4	LPMRES	R	如果MH1902T因任何原因被恢复时该位被设置。该位与电源寄存器（地址0x01）的RESUME位互斥
D3	LPMNC	R	当接收到LPM事务和MUSBMDHRC由于数据在RX FIFO的等待没有回应时该位被设置，在以下条件下才会发生： 在LMSRESP寄存器的LPMRESP字段设置为2'b11，所述LPMXMT字段设置位1'b1并有数据待决的MH1902T TX FIFO中。
D2	LPMACK	R	该位在接收到LPM事务和MUSBMDHRC产生一个ACK时被设置，这在以下条件下才会发生： 在LMSRESP寄存器的LPMRESP字段设置为2'b11，所述LPMXMT字段设置位1'b1并有数据待决的MH1902T TX FIFO中。
D1	LPMNY	R	该位在接收到LPM事务时和MUSBMDHRC响应一个NYET时被设置。这在以下情况中才会发生： 在LMRESP寄存器的LPMRESP字段设置为2'b11和LPMXMT字段设置位1'b0
D0	LPMST	R	该位在接收到LPM事务时和MUSBMDHRC响应一个STALL时被设置，这在以下条件下才会发生： 在LMRESP寄存器的LPMRESP字段设置为2'b01

在主机模式下：

D7	D6	D5	D4	D3	D2	D1	D0
RESERVED			LPMRES	LPMNC	LPMNCK	LPMNY	LPMST
r	r	r	r	r	r	r	r

Bit	Name	W/R	Description
D7-D6	Reserved	R	预留，读取时总是返回0x0
D5	LPMERR	R	该位在收到带有位填充错误或者PID误差的LPM事务响应时被设置。在这种情况下，不能暂停且该装置的状态未可知
D4	LPMRES	R	该位在MH1902T以任何原因恢复时被设置。该位与电源寄存器（地址0x01）的RESUME位互斥
D3	LPMNC	R	该位在LPM事务已发送或发送失败时被设置，发生超时或有三个尝试响应错误时此次交易将失败
D2	LPMACK	R	该位在LPM事务被发送和设备响应一个ACK时被设置
D1	LPMNY	R	该位在LPM事务被发送和设备响应一个NYET时被设置
D0	LPMST	R	该位在LPM事务被发送和设备响应一个STALL时被设置

LPM_FADDR

初步认识：仅在主机模式

偏移地址：0x365 复位值：0x00

LPM_FADDR是函数地址将被放置在LPM有效载荷

D7	D6	D5	D4	D3	D2	D1	D0
0	Function Address						
r	rw	rw	rw	rw	rw	rw	rw

Bit	Name	W/R	Description
D7	-	R	未使用，始终返回0

D6-D0	LPM FADDR	W/R	该LPM函数地址

19.5 USB复位

19.5.1 外设模式下

当MH1902T充当外围设备和检测到USB上其他重置条件时，该装置将执行以下操作：

- 设置 FADDR 为 0
- 设置索引为 0
- 刷新所有端点的 FIFO
- 清除所有控制/状态寄存器
- 启用所有的端点中断
- 产生一个复位中断

当应用软件驱动MH1902T收到一个复位中断，应该关闭所有打开的管道并等待总线枚举的开始。

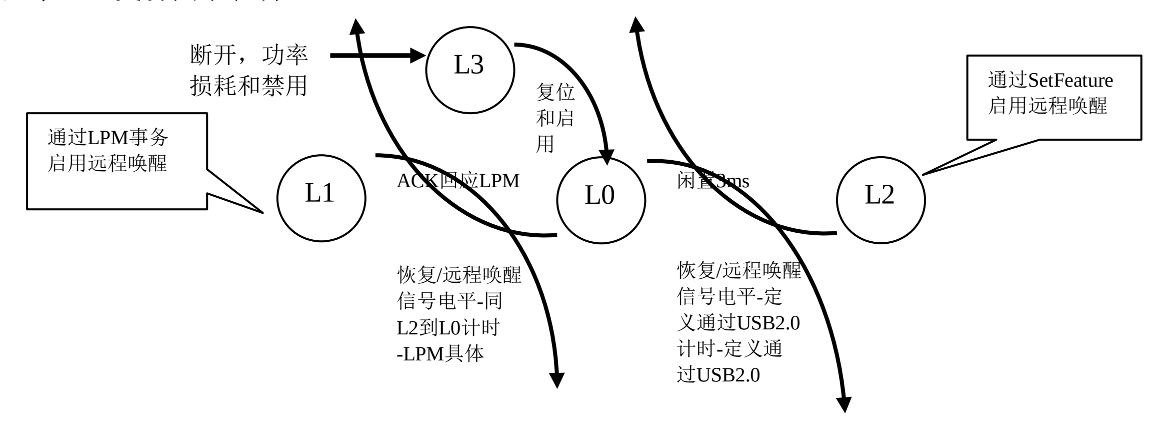
19.5.2 主机模式下

如果电源寄存器中的复位位被设置，而MH1902T是主机模式时MH1902T将产生总线上的复位信号。

该CPU应该保持复位位设置至少20毫秒，以保证目标设备的正确复位。CPU清除该位后，MH1902T将开始帧计数和事务调度。

19.6 暂停/恢复

通过引入连接电源管理，有两个基本方法用于MH1902T暂停和恢复。这两个方法都在如下所示的基本LPM交易图中表明



其中芯片被挂起和恢复的过程取决于核心是否作为设备或主机操作和刮起所需要的方法。

19.6.1 芯片作为外设运行

1) 进入到暂停模式。当作为外设工作时，芯片监控 USB 活动，3 毫秒没有活动发生时进入挂起模式。如果暂停中断已被允许，此时将产生中断，该 SUSPENDM 输出也变低（如果启用）。
在这一点上，POWERDWN信号也被认定，以表明该应用可以节省电能通过停止CLK。
POWERDWN一直保持有效，直到电源从总线移除（表示该设备已断开连接）或恢复信号或复位信号检测到总线上。
2) 当恢复信号在总线上产生时，第一个 CLK 必须在必要时重新启动。然后芯片会自动退出暂停模式。如果恢复中断使能，中断将产生
3) 启动远程唤醒。如果软件要启动远程唤醒，而芯片在暂停模式下，它应该写入功率寄存器设置恢复位（D2）为“1”。（注意：如果 CLK 已经停止，则需要重新启动，在此之前可能就要写入）。软件重置前应预留该位重置为 0 前的设置大概 10 毫秒（最小为 2 毫秒，最大为 15 毫秒）。在这

个时候枢纽应该接管 USB 上的运行信号

注：当软件启动远程唤醒没有恢复时，中断将产生。

19.6.2 芯片作为主机运行

1) 进入到暂停模式。当作为主机时，芯片可以通过电源寄存器中的挂起模式位提示进入挂起模式。当电源寄存器的挂起模式位被设置时，芯片将完成当前事务，停止事务调度和帧计数，没有进一步的事务被启动，并且没有 SOF 包生成
   如果启用 SuspendM 位（Power.D0）设置，该 UTML+PHY 将进入低功耗模式下，当芯片进入挂起模式，停止 XCLK。
2) 发送恢复信号。当应用程序需要芯片离开挂起模式时，它需要清除挂起位，设置电源寄存器的恢复位并保持该位设置 20 毫秒。恢复位高，芯片将会产生恢复总线信号。经过 20 毫秒，CPU 明确恢复位，此时帧计数和事务调度程序将被启动。
3) 应对远程唤醒。如果从目标检测到恢复信号而芯片处于挂起模式时，UTML+PHY 将被带出低功率模式，重新启动 XCLK。该 MH1902TS 退出挂起模式并自动设置在电源寄存器（D2）为“1”，恢复位从目标接管信号产生恢复。如果恢复中断使能，中断将产生。

19.7 连接/断开

相关连接和断开芯片的特定行为无论是在主机模式下还是在对等通信外设模式中都可以使用。

19.7.1 主机模式下

芯片在主机模式下时，CPU通过设置会话位（DecCtl.D0）启动会话。电能被应用到VBUS，芯等待要连接的装置。

当检测到一个装置时，一个连接中断产生（即IntrUSB.D4变高）。已连接设备的速度可通过读取DevCtl寄存器来确定。FSDev（D6）会为全速设备变高，Isdev（D5）会为低速设备变高。之后CPU重置该设备，如果FSDev和HS ENAB（Power.D5）同时设置，芯片将尝试为高速运行进行谈判，是否成功使用高速模式位（Power.D4）表示。

该CPU应该保持复位位设置20毫秒，以确保目标复位，之后可以开始设备枚举。

如果设备断开连接，而会话正在进行，一个断开将会产生中断（即IntrlUS.D5变高）。

19.7.2 外设模式下

其中芯片在外设模式下操作时，该设备被连接到主机上不产生中断。然而当主机中止会话时一个断开中断（IntrlUSB.D5）产生。

19.8 规划方案

这与以下各部分看，该装置控制所述芯片需要执行的操作与在该影响下核心操作的各个方面。

在整个讨论中，控制装置被假定为运行某些固件的单片机，但它可以定制硬布线逻辑块。

19.8.1 软连接/断开

如果需要的话，该芯片将允许其连接到由软件控制的USB总线上。

当软连接/断开被选中，那么当芯片工作在外设模式下时，该UTMI+-一起使用芯片标准的PHY可以在正常模式与非驱动模式之间通过设置/清除电源寄存器的第6位（切换被确定为软连接位）。当该软连接位设置为1时，所述的PHY被放置在其正常模式和被启用USB总线的D+/D-线。在同一时间，该芯片被放置在“供电”状态，将不再回应任何USB信号，除USB重置。

如果启用此功能，软连接位为0，物理层被置于非驱动模式下，D+和D-为3态，如果它已经断开，芯片将在USB总线上出现其它设备。

硬件复位（NRST=0）后，软连接被清除为0。芯片将因此出现短线，直到软件设置软连接位为

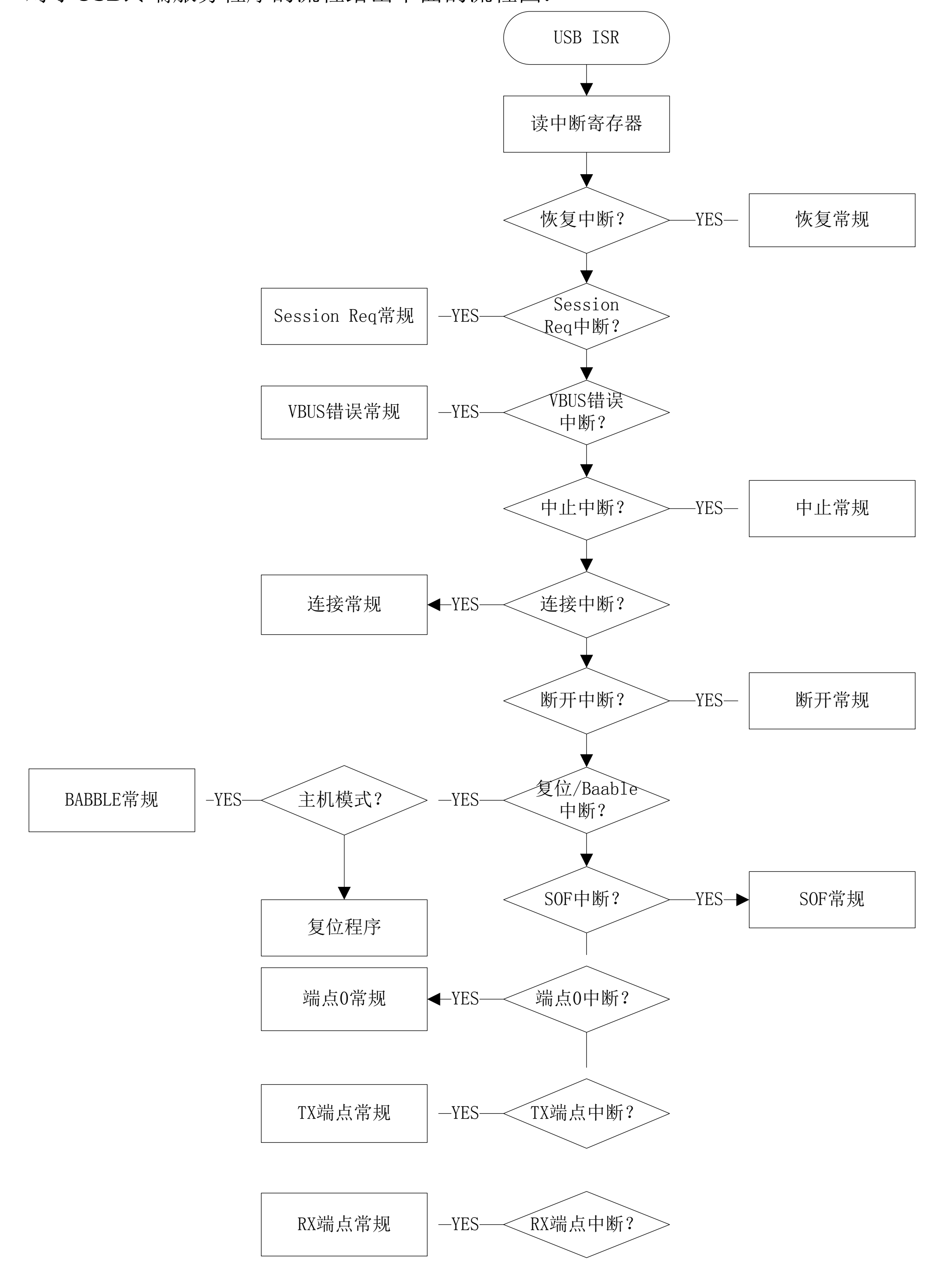
1。之后，应用软件可以选择何时将PHY设置到正常模式。冗长初始化过程可能会借此确保初初始化系统完成，系统已准备好在连接到USB之前执行枚举。

一旦软连接位已被设置为1，该软件还可以通过清除此位模拟脱节。

19.8.2 USB中断处理

当在有一个USB中断CPU断开时，需要读取中断状态寄存器来确定是哪个端点造成的中断，并跳转到相应的程序。如果是多个端点造成的中断，端点0先服务，其次是其他终端。

对于USB终端服务程序的流程给出下面的流程图：



19.9 OTG会话请求

为了节省电力，在USB规范上允许VBUS仅在需要时通电，在不使用时中断通电。该芯片进一

步通过允许CLK在没有会话正在进行时停止工作提升了额外的省电。当POWERDWN被断言时，CLK可能会被停止。

VBUS总是由总线上的“A”设备提供。该芯片通过从UTML+PHY的IDDIG采样来确定它是否是“A”设备或“B”设备。这个信号被拉低时，一个A型插件被感测（表示该芯片是A设备），信号采取高时，B型插件被感测（表示该芯片是B设备）。

19.9.1 启动会话

当含有芯片设备需要启动一个会话时，CPU需要在DevCtl寄存器（D0）设置会话位。该芯片会启用ID引脚检测。这导致IDDIG在检测到A型连接时输出低，在检测到B型连接时输出高。在DevCtl寄存器（D7）B设备位也被设置，起到指示芯片是否已通过了“A”设备或“B”设备的作用。

如果芯片是A设备：芯片将进入主机模式（A设备始终默认主机），打开VBUS并等待VBUS达到有效的VBUS阈值时，由VBUSVALID指示输入信号上升。（该事件还是要VBUS[1:0]中的寄存器DevCtl（D4位-D3）达到11b）

芯片等待要连接的外设。当检测到外设，一个连接中断（IntrUSB.D4）将产生（如果使能），FSDev或在DevCtl寄存器的Isdev（D6/D5）被设置，这取决于外设是全速外设还是低速外设。

CPU重置该外设。如果FSDev或在DevCtl寄存器的Isdev（D6/D5）同时被设置，芯片将复位监视LINESTATE，看是否有从外围接收的信号。如果收到信号，该芯片响应该信号，并进入该模式。

结束会话。要结束会话，该CPU应清楚会话位（DevCtl.D0）。如果检测到路重合，该芯片也会自动终止会话。

如果芯片是B设备：芯片将使用在USB规范山上的会话请求协议，也就是说它会先断言DISCHRGVBUS发出的VBUS会话请求。之后当VBUS低于会话结束阈值（如由SESSEND输出使表示高，则VBUS[1:0]在DevCtl寄存器位（D4-D3）为00b），线路状态一直为SE0>2毫秒，芯片将首先脉冲数据线，然后脉冲VBUS（通过CHRGVBUS取高）。

在会话结束时，会话位清0。通常由芯片来执行，也可以通过CPU清零，如果应用软件希望执行一个软件中断，芯片将切换到TERMSEL，这将导致PHY转出的D+上拉电阻。这标志着在A设备结束会话。

19.9.2 检测活动

当OTG建立会话启动其他设备时，如果是A设备，它会提高VBUS使之高于会话有效的阈值（由AVALID输入上升信号和VBUS[1:0]在DevCtl寄存器位（D4-D3）为10b）或者如果它是B设备将首先脉冲数据线，然后脉冲VBUS。芯片根据这些情况的发生来确定当前设置是A设备还是B设备。详情如下：

如果VBUS高于会话有效值：那么芯片是B设备。该芯片将设置DevCtl寄存器位（D0）会话位。当总线上检测到复位信号时将产生复位中断（如果使能）。其中CPU对此解释为一个会话的开始，在外设模式下芯片将默认外设时B设备。

在会话结束时，A设备将关闭电源VBUS。当VBUS低于会话有效的阈值（AVALID输入指示变低，VBUS[1:0]在DevCtl寄存器（D4-D3）将变为01B）。芯片将检测到这些变化并清除会话位来指示会话已结束，产生断开连接中断（IntrUSB.D5如果使能）。

如果数据线/USB检测到脉冲，则芯片是A设备。它会生成一个会话请求中断（IntrUSB.D6-如果启用），来表明B设备请求会话，之后CPU应启动一个会话。

19.10 主机协商

当芯片是’A’的设备（IDDIG低，B设备（DevCtl.D7）=0）会话启动时，它会自动进入主机模式。

当芯片是’B’的设备（IDDIG高，B超设备（DevCtl.D7）=1）会话启动时，它会自动进入外设模式。然而，CPU可以请求芯片成为主机通过设置DevCtl寄存器（D1）。该位可设置或者在同时请求会话开始通过设置会话位（DevCtl.D0），或在会话开始的任何时候。当芯片接下来进入挂起模式（总线上3毫秒没有活动），则假设HOST Req位保持设置，将进入主机模式，并开始主机协商（如USB的On-The –Go补充规定）切换至TERMSEL，导致所述PHY断开在D+线的上拉电阻。这应该会导致“A”的设备切换到外设模式，连接自己的上拉电阻。当芯片检测到启用了该选项，它会生成一个连接中

断（IntrUSB.D4）。它也将设置在电源寄存器（D3）复位位开始复位’A’设备。（该芯片自动开始这个复位序列，确保复位启动内1毫秒’A’设备连接其上拉电阻的要求）。该CPU至少应当等待20毫秒，然后清除复位位，并列举“A”设备。

当芯片为基础的“B”设备使用总线完成时，CPU应该通过设置电源寄存器（D1）挂起模式位把它进入挂起模式。在“A”的设备应该检测到这个，要么终止会话或恢复到主机模式。如果“A”设备芯片为基础启用该选项，它会生成一个断开连接中断（IntrUSB.D5）。

19.11 VBUS活动

USB规范定义了一系列涉及在点对点通信设备需要对应的阈值：

- VBUS 有效（要求在 4.4 和 4.75）
- 会话有效期为’A’设备（要求在 0.8V 和 2.1V 之间）
- 会话结束（要求在 0.2V 和 0.8V 之间）

（在一特定装置中使用的实际阈值需要通过一系列比较器，外部MH1902TS核心。外部MH1902TS核心采取相应的VBUSVALID，AVALID和SESSEND依据VBUS级别设置输入高或低）

其中这些阈值是关键的，其中CPU控制MH1902TS需要响应的方式取决于设备是’A’设备或’B’设备和发生其他事件的情况。所需操作总结如下：

19.11.1作为’A’设备操作

1) VBUS>VBUS 由 MH1902TS 启动的会话有效值（VBUS[1:0]与 DevCtl[D4:D3]=11B，会话位（DevCtl.D0）设置)。当 VBUS 比 VBUS 有效值大时，MH1902TS 选择主机模式和等待要连接的装置。它会生成一个连接中断（IntrUSB.D4)。CPU 重置并列举连接的’B’设备。
2) VBUS>VBUS 由’B’设备启动的会话有效值（VBUS[1:0]与 DevCtl[D4:D3]=10B，会话位（DevCtl.D0）设置)。当 VBUS 比 VBUS 有效值大时，MH1902TS 会产生一个会话请求中断（IntrUSB.D6)。CPU 设置会话位，MH1902TS 要么留在主机模式，要么根据’B’设备上拉电阻状态改变为外设模式。所选择的模式将由主机模式位（DevCtl.D2）的状态来表示。
3) VBUS 低于 VBUS 会话保持有效值（VBUS[1:0]与 DevCtl[D4:D3]不等于 11B，会话位（DevCtl.D0）设置)。这表明 VBUS 功率电平问题，例如：电池电量可能已下降过低来维持 VBUS 有效。另外，在’B’设备可以吸引比’A’设备可以提供的更多的电流。在这两种情况下，芯片将会自动终止会话，并生成一个 VBUS 错误中断（IntrUSB.D7)。

19.11.2作为’B’设备操作

1）VBUS>会话有效值（即 Vbus[1:0](DevCTL[D4:D3])=10B，会话位（DevCtl.D0）设置)。这表明从“A”设备活动。芯片将设置会话位，并采取 DPPULLDOWN 输出低电平来断开 D+线下拉电阻。
2）VBUS<会话有效值，而会话位保持设置（即 Vbus[1:0](DevCTL[D4:D3])=01B，会话位（DevCtl.D0）设置)。这表明“A”设备已经失去权力（或断开连接）。芯片将清除会话位（DevCtl.D0)，并产生一个断开中断（IntrUSB.D5)。CPU 结束会话。
3）VBUS<会话有效值（即 Vbus[1:0](DevCTL[D4:D3])=00B)。这是下一个“B”设备可以发起会话请求的条件。如果会话位（DevCtl.D0）设置，SE0 在总线上执行 2 毫秒之后，芯片将首先脉冲数据线，然后脉冲 VBUS（采取 CHRGVBUS 高点）开始调整计划。

19.12 动态FIFO

如果需要的话，芯片可被构造成具有128,256,512…1K 64K字节、分区大小的的FIFO，当芯片初始化时可继续被分配给不同的端点。

注：我们强烈建议您仅使用此功能，其中芯片用在在不同环境下需要不同FIFO大小的设备。如果FIFO大小不需要改变，最好是通过标准配置选项来设置这些尺寸为动态FIFO尺寸的选项来显著增加核心的尺寸，并且需要更复杂的固件来处理它。

FIFO根据每个TX和RX端点的不同要求规范分配的空间：

- FIFO 的 RAM 块的起始地址

- 要支持数据包的最大尺寸
- 双缓冲是否是必需的

（最后这两个共同限定的空间需要被分配给 FIFO）

这些细节可以通过下列四个寄存器来指定，当动态FIFO大小选项被指定时它们被加入到芯片寄存器映射的索引区域：

ADDR	Name	Description
62	TxFIFOsz	Tx端点FIFO大小
63	RxFIFOsz	Rx端点FIFO大小
64,65	TxFIFOadd	Tx端点FIFO地址
66,67	RxFIFOadd	Rx端点FIFO地址

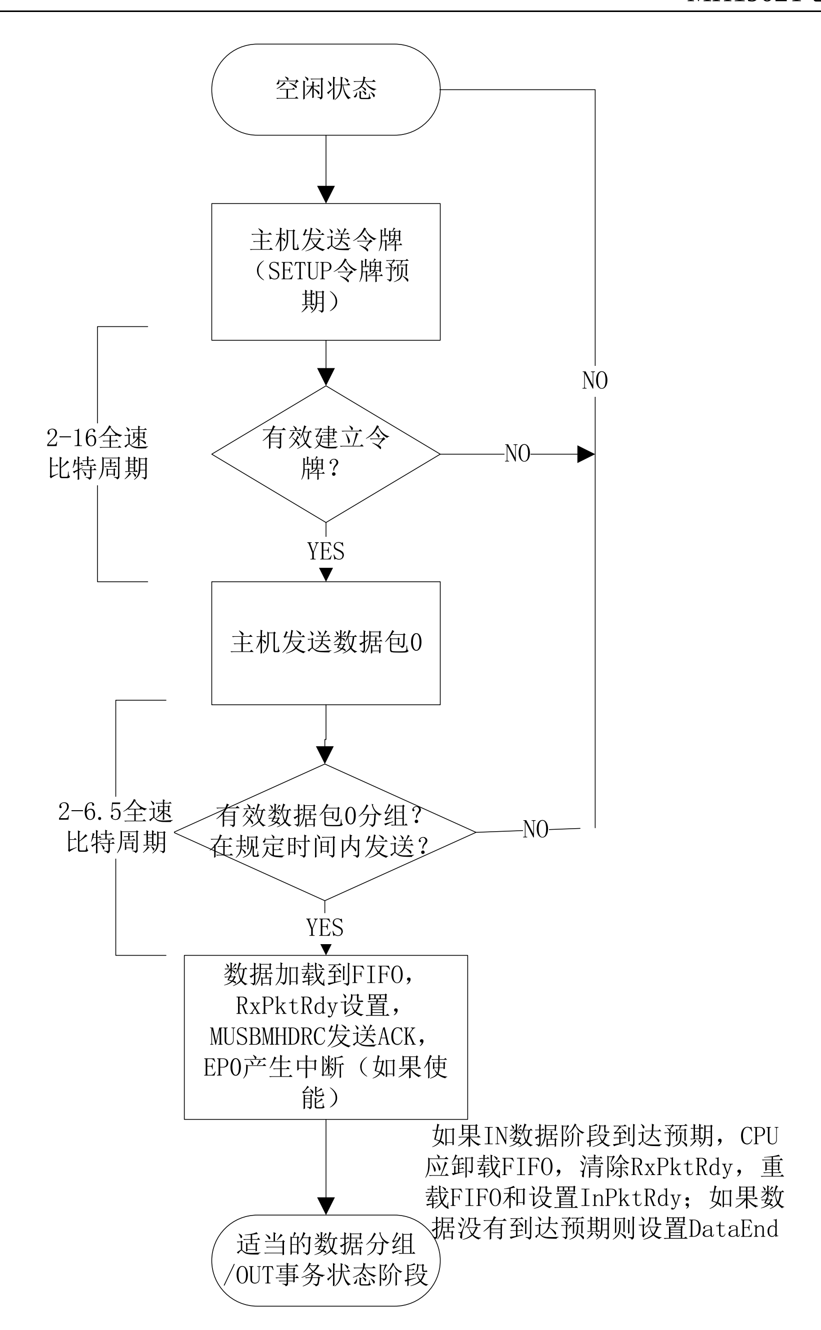
注：（i） 设定动态FIFO的选项尺寸仅适用端点1-15。FIFO的端点0有一个固定的大小（64字节）和一个固定的位置（起始地址0）

（ii） 它是固件的责任（和系统设计者）。来确保所有的发射和接收终端处于活动状态。当前USB配置的RAM块分配给它们至少和端点设置最大数据包大小一样大。

19.13 事务流作为外设

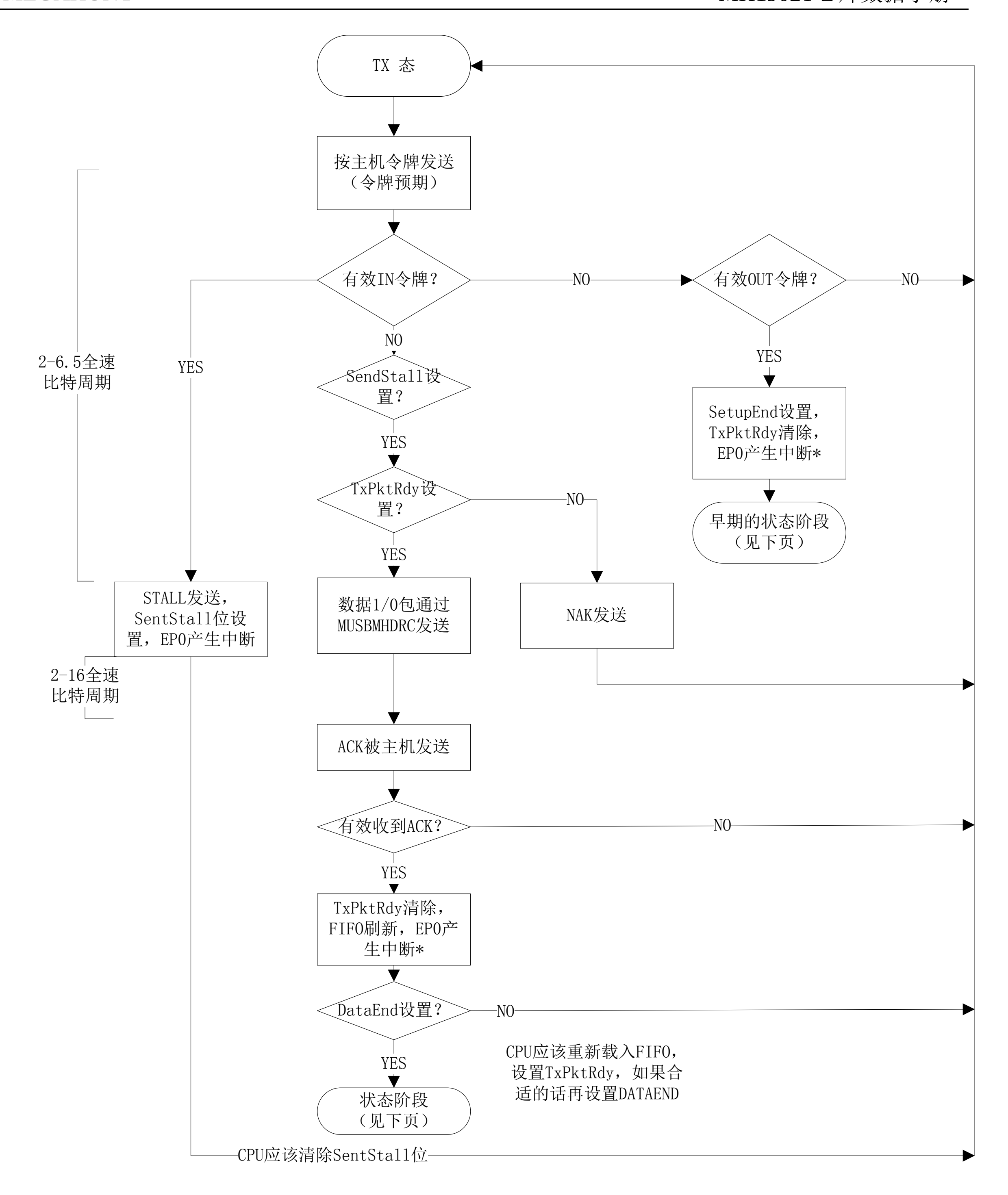
19.13.1控制事务

安装阶段

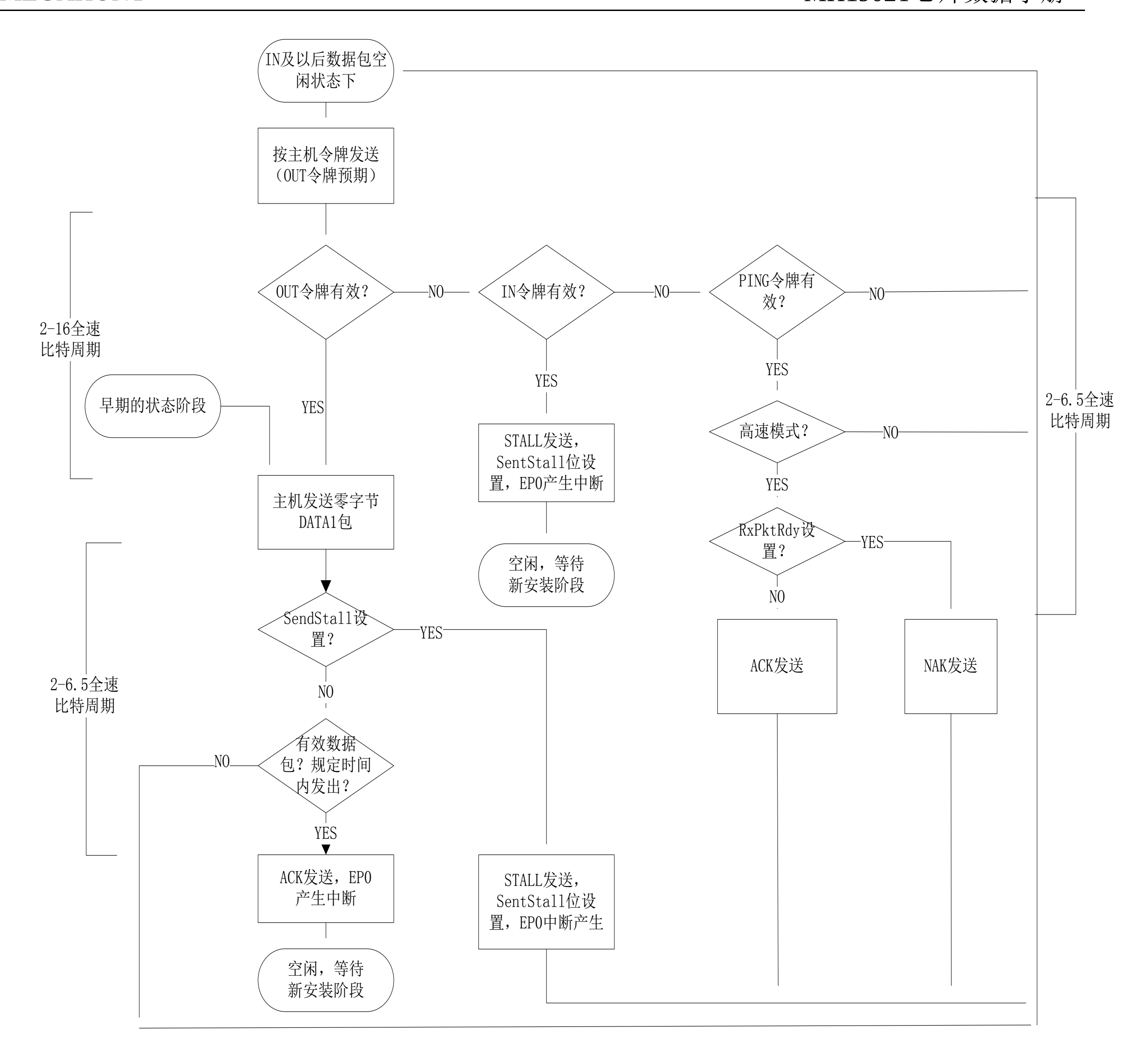


空闲状态
主机发送令牌（SETUP令牌预期）
NO
2-16全速比特周期
有效建立令牌？
NO
YES
主机发送数据包0
2-6.5全速比特周期
有效数据包0分组？在规定时间内发送？
NO
YES
数据加载到FIFO，RxPktRdy设置，MUSBMHDRC发送ACK，EP0产生中断（如果使能）
如果IN数据阶段到达预期，CPU应卸载FIFO，清除RxPktRdy，重载FIFO和设置InPktRdy；如果数据没有到达预期则设置DataEnd
适当的数据分组/OUT事务状态阶段

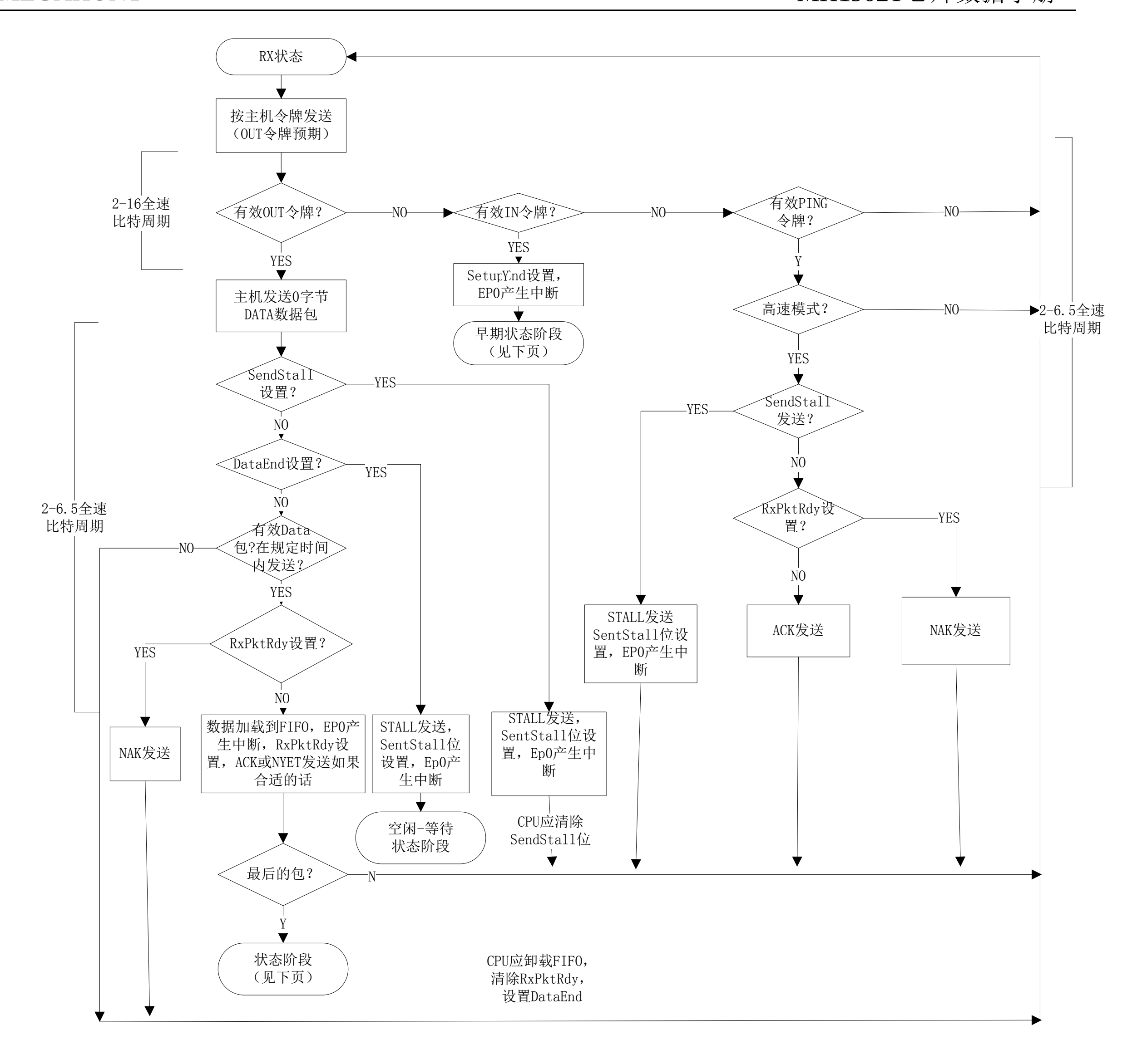
数据图


TX 态
按主机令牌发送
（令牌预期）
有效IN令牌？
NO
有效OUT令牌？
NO
2-6.5全速
比特周期
YES
NO
SendStall设
置？
YES
SetupEnd设置，
TxPktRdy清除，
EP0产生中断*
YES
TxPktRdy设
置？
NO
早期的状态阶段
（见下页）
YES
STALL发送，
SentStall位设
置，EP0产生中断
数据1/0包通过
MUSBMHDRC发送
NAK发送
2-16全速
比特周期
ACK被主机发送
有效收到ACK？
NO
YES
TxPktRdy清除，
FIFO刷新，EP0产
生中断*
DataEnd设置？
NO
CPU应该重新载入FIFO，
设置TxPktRdy，如果合
适的话再设置DATAEND
YES
状态阶段
（见下页）
CPU应该清除SentStall位

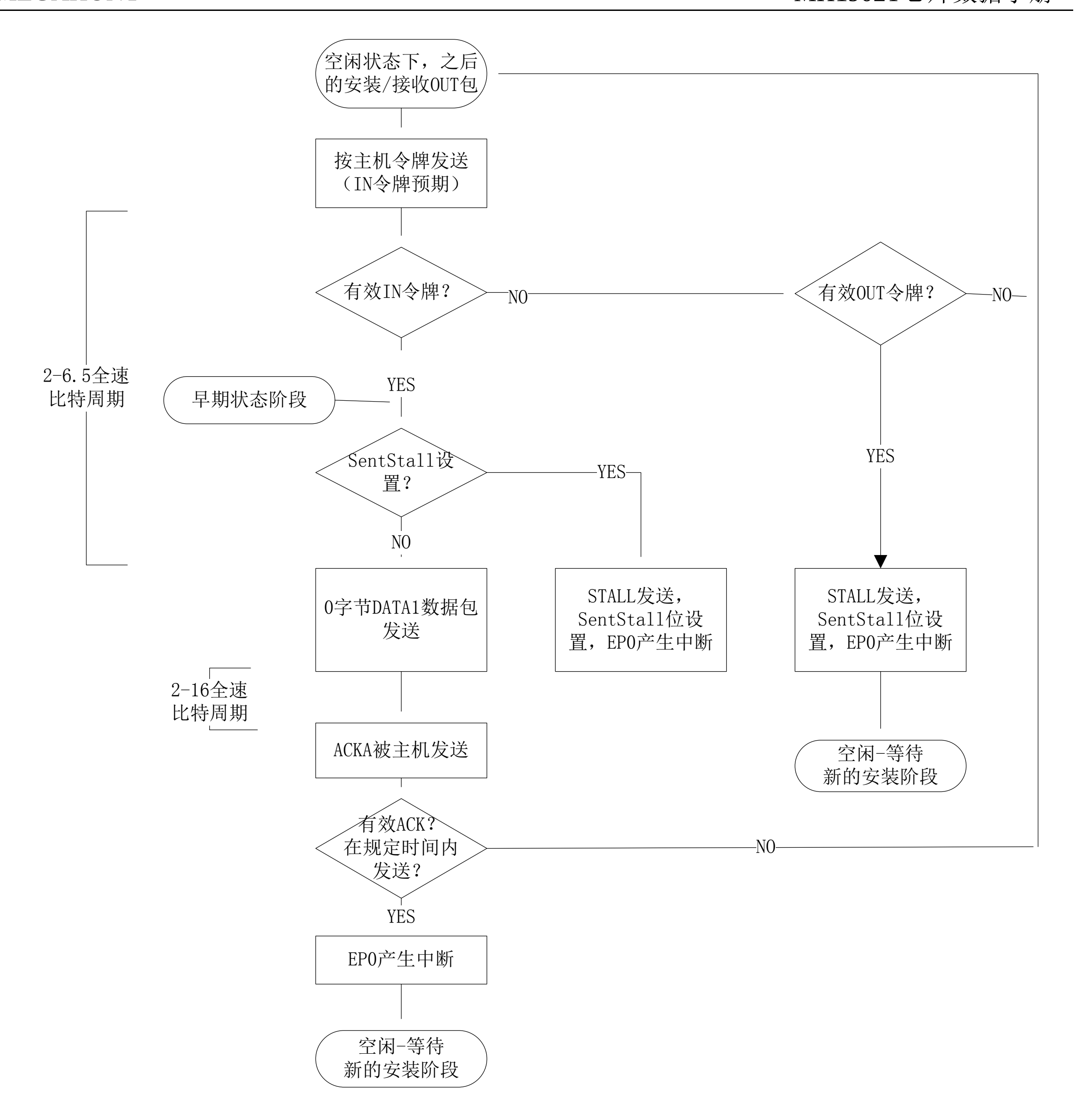
之后的状态阶段


IN及以后数据包空闲状态下
按主机令牌发送（OUT令牌预期）
OUT令牌有效？
NO
IN令牌有效？
NO
PING令牌有效？
NO
2-16全速比特周期
早期的状态阶段
YES
YES
STALL发送，SentStall位设置，EP0产生中断
YES
高速模式？
NO
2-6.5全速比特周期
YES
主机发送零字节DATA1包
空闲，等待新安装阶段
RxPktRdy设置？
YES
NO
SendStall设置？
YES
ACK发送
NAK发送
NO
2-6.5全速比特周期
有效数据包？规定时间内发出？
NO
YES
ACK发送，EP0产生中断
STALL发送，SentStall位设置，EP0中断产生
空闲，等待新安装阶段

OUT数据状态

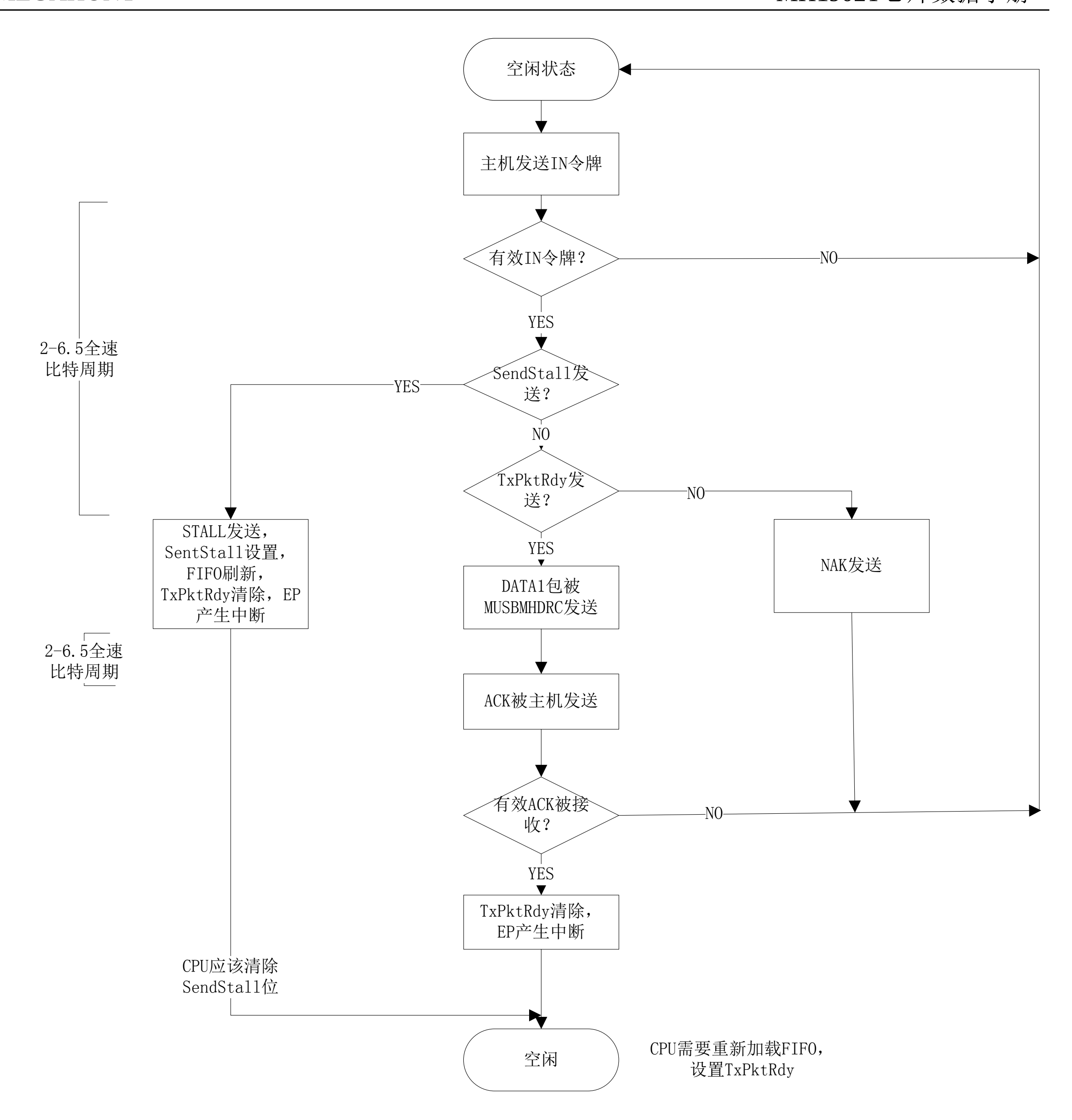

RX状态
按主机令牌发送（OUT令牌预期）
2-16全速比特周期
有效OUT令牌？
NO
有效IN令牌？
NO
有效PING令牌？
NO
YES
SetupEnd设置，EP0产生中断
Y
高速模式？
NO
主机发送0字节DATA数据包
早期状态阶段（见下页）
2-6.5全速比特周期
YES
SendStall设置？
YES
SendStall发送？
YES
NO
NO
DataEnd设置？
YES
2-6.5全速比特周期
NO
RxPktRdy设置？
YES
有效Data包?在规定时间内发送？
NO
YES
NO
RxPktRdy设置？
YES
STALL发送SentStall位设置，EP0产生中断
ACK发送
NAK发送
NO
NAK发送
数据加载到FIFO，EP0产生中断，RxPktRdy设置，ACK或NYET发送如果合适的话
STALL发送，SentStall位设置，Ep0产生中断
STALL发送，SentStall位设置，Ep0产生中断
空闲-等待状态阶段
CPU应清除SendStall位
最后的包？
N
Y
状态阶段（见下页）
CPU应卸载FIFO，清除RxPktRdy，设置DataEnd

之后的状态阶段

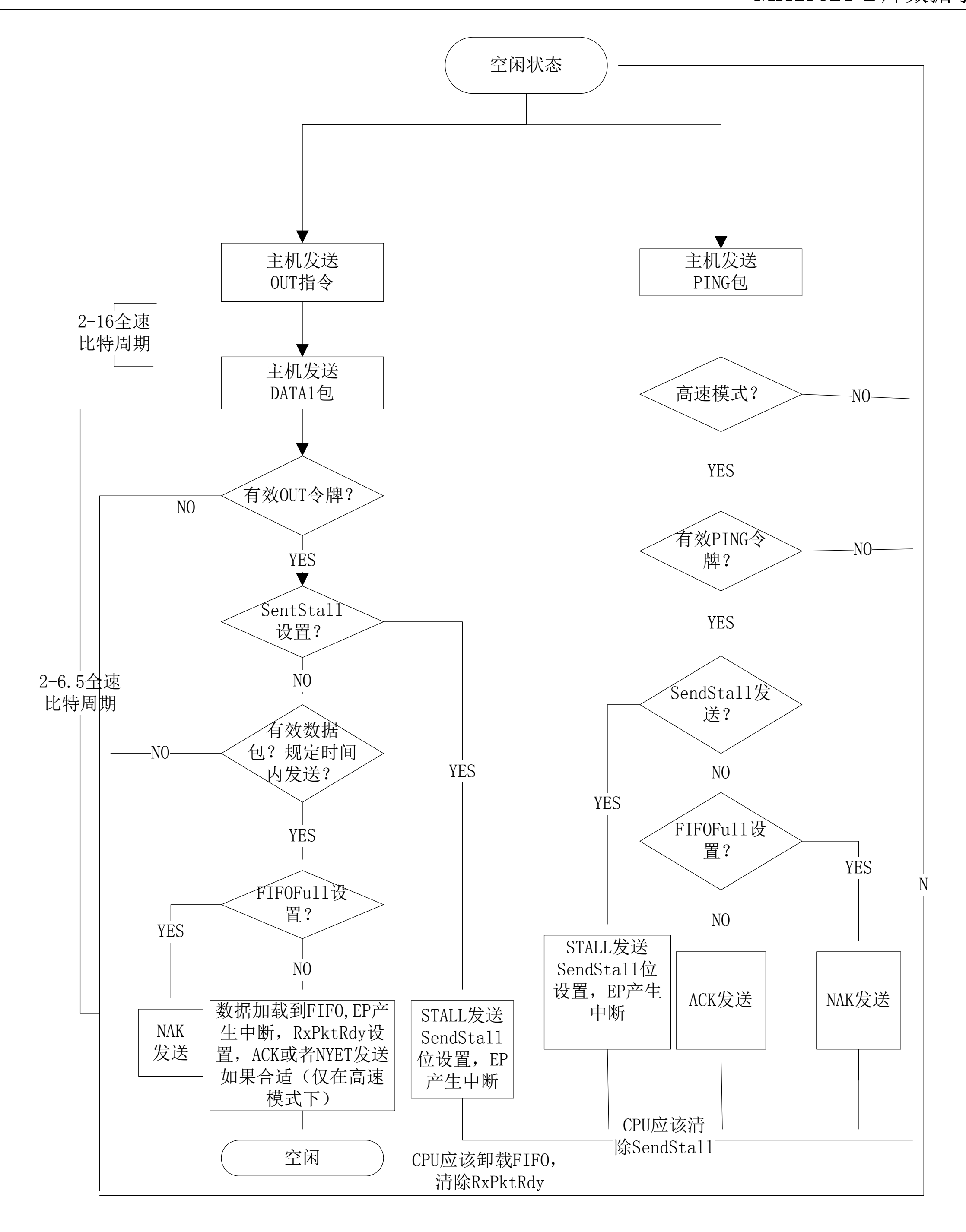


19.13.2BULK/低带宽中断事务

IN事务

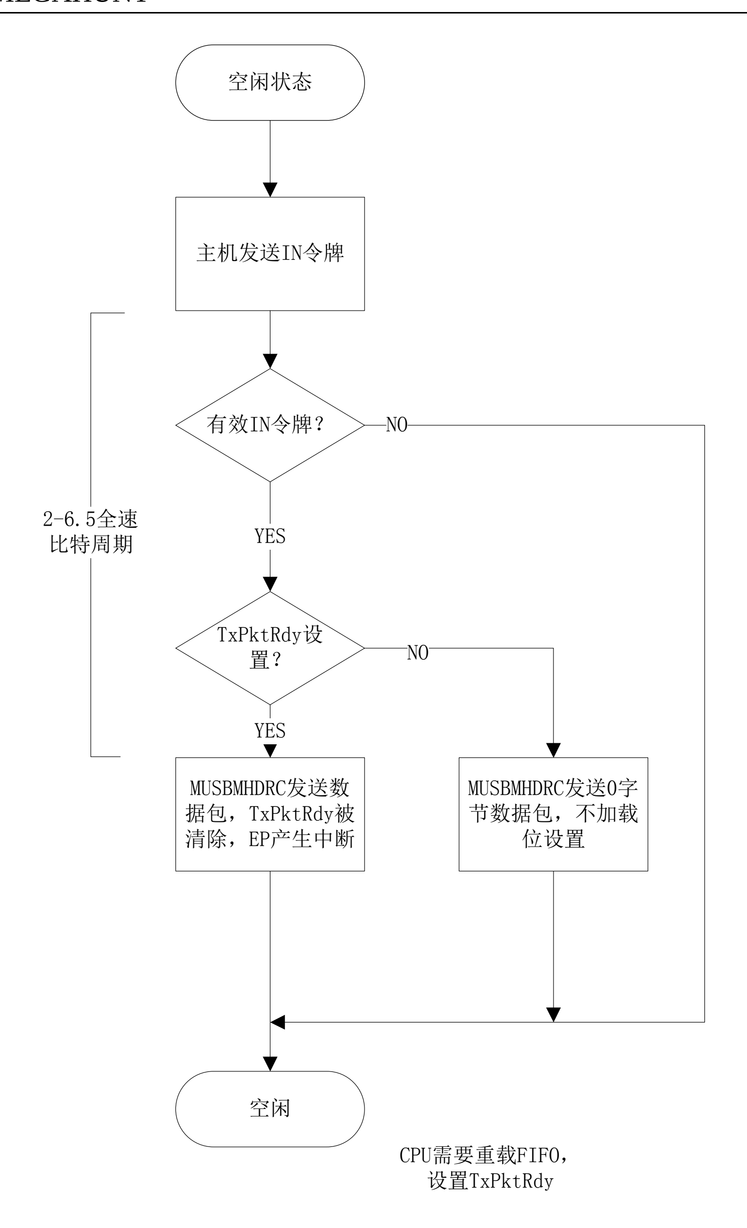

空闲状态
主机发送IN令牌
有效IN令牌？
NO
YES
2-6.5全速
比特周期
SendStall发
送？
YES
NO
TxPktRdy发
送？
NO
YES
STALL发送，
SentStall设置，
FIFO刷新，
TxPktRdy清除，EP
产生中断
NAK发送
DATA1包被
MUSBMHDRC发送
2-6.5全速
比特周期
ACK被主机发送
有效ACK被接
收？
NO
YES
TxPktRdy清除，
EP产生中断
CPU应该清除
SendStall位
空闲
CPU需要重新加载FIFO，
设置TxPktRdy

OUT事务

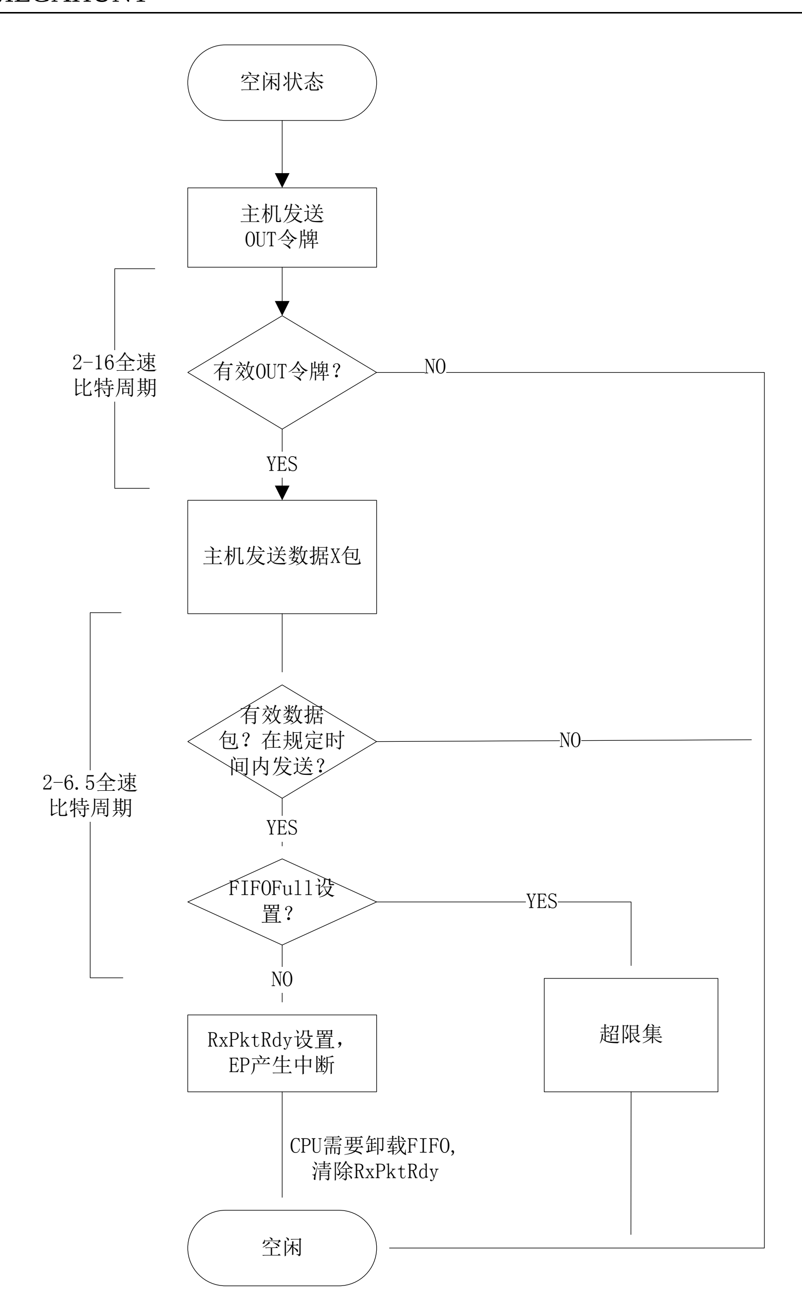


19.13.3全速/低带宽等时事务

IN事务

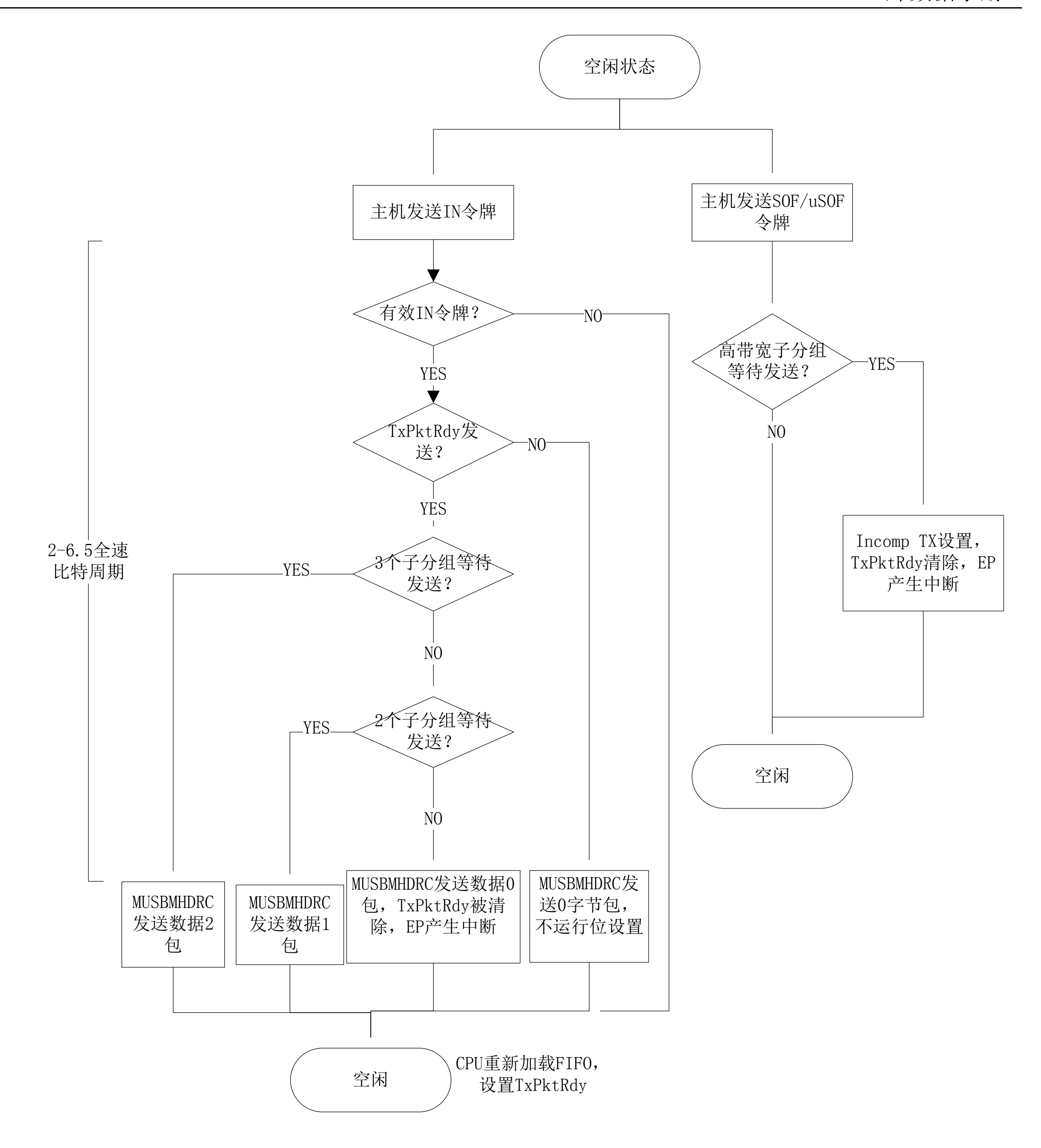

空闲状态
主机发送IN令牌
有效IN令牌？
NO
YES
2-6.5全速
比特周期
TxPktRdy设
置？
NO
YES
MUSBMHDRC发送数
据包，TxPktRdy被
清除，EP产生中断
MUSBMHDRC发送0字
节数据包，不加载
位设置
空闲
CPU需要重载FIFO，
设置TxPktRdy

OUT事务

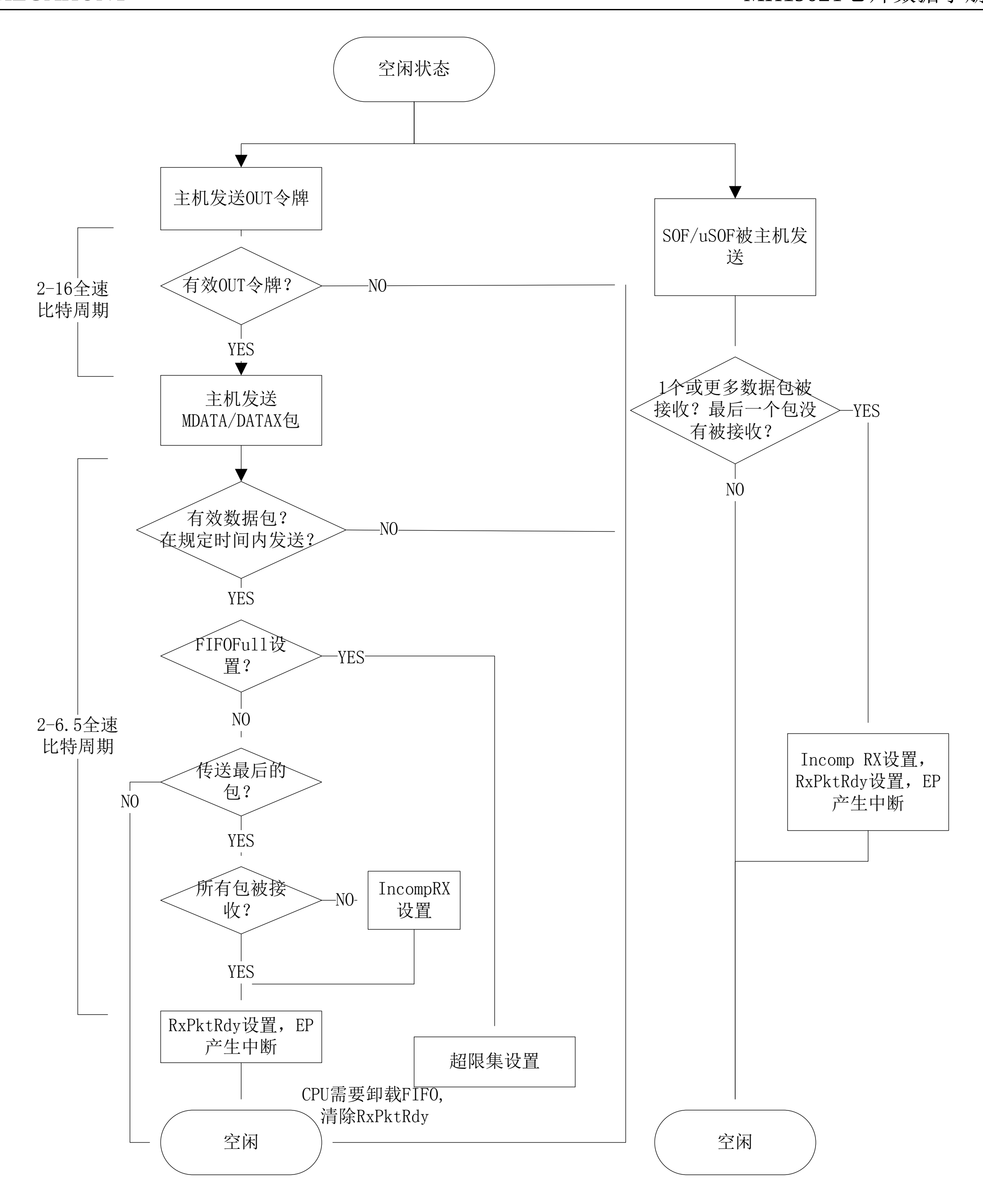


19.13.4高带宽事务（同步/中断）

IN事务


空闲状态
主机发送IN令牌
主机发送SOF/uSOF令牌
有效IN令牌？
NO
YES
高带宽子分组等待发送？
YES
NO
TxPktRdy发送？
NO
YES
Incomp TX设置，TxPktRdy清除，EP产生中断
2-6.5全速比特周期
YES
3个子分组等待发送？
NO
空闲
YES
2个子分组等待发送？
NO
MUSBMHDRC发送数据2包
MUSBMHDRC发送数据1包
MUSBMHDRC发送数据0包，TxPktRdy被清除，EP产生中断
MUSBMHDRC发送0字节包，不运行位设置
空闲
CPU重新加载FIFO，设置TxPktRdy

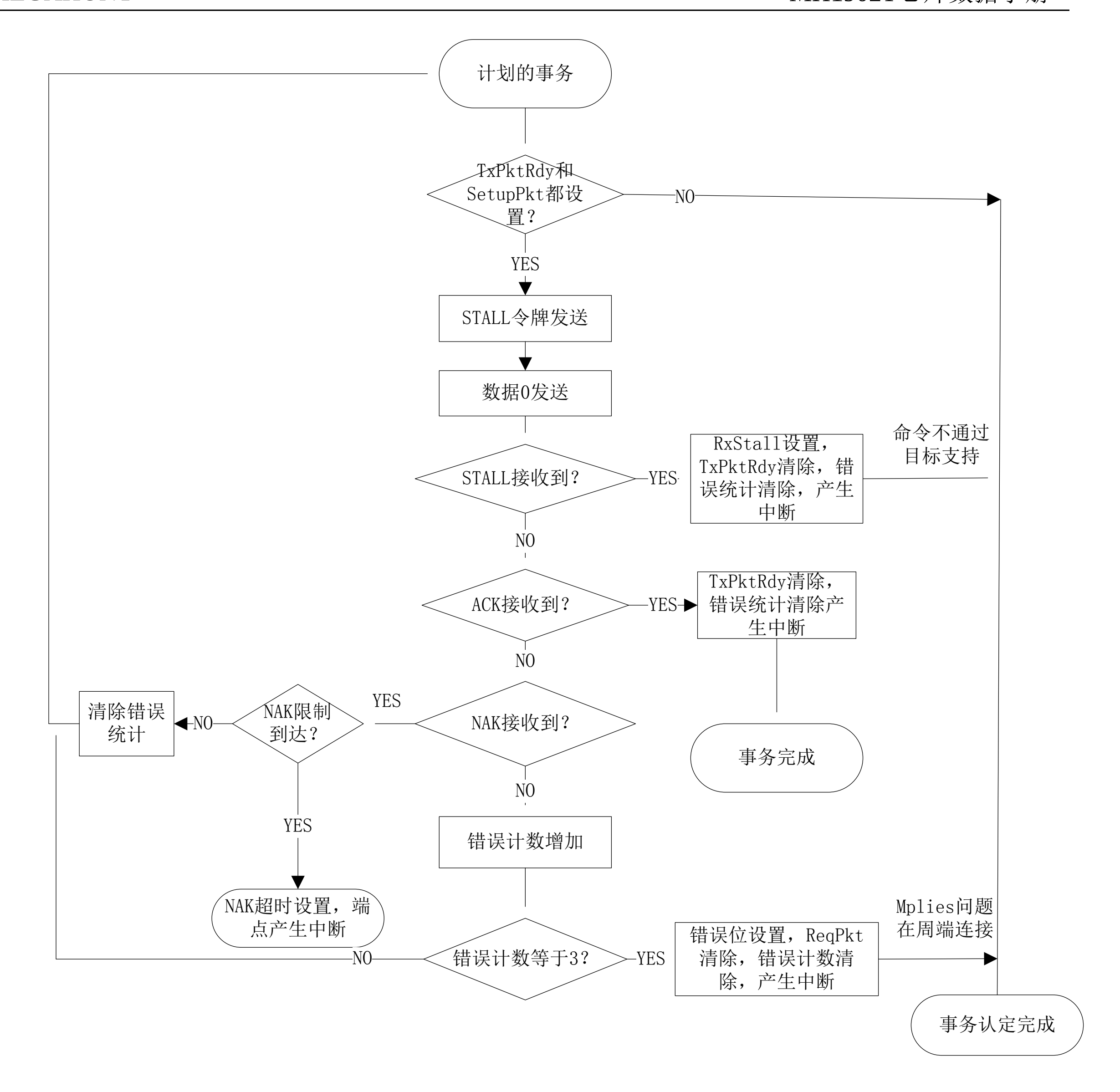
OUT事务



19.14 事务流作为主机

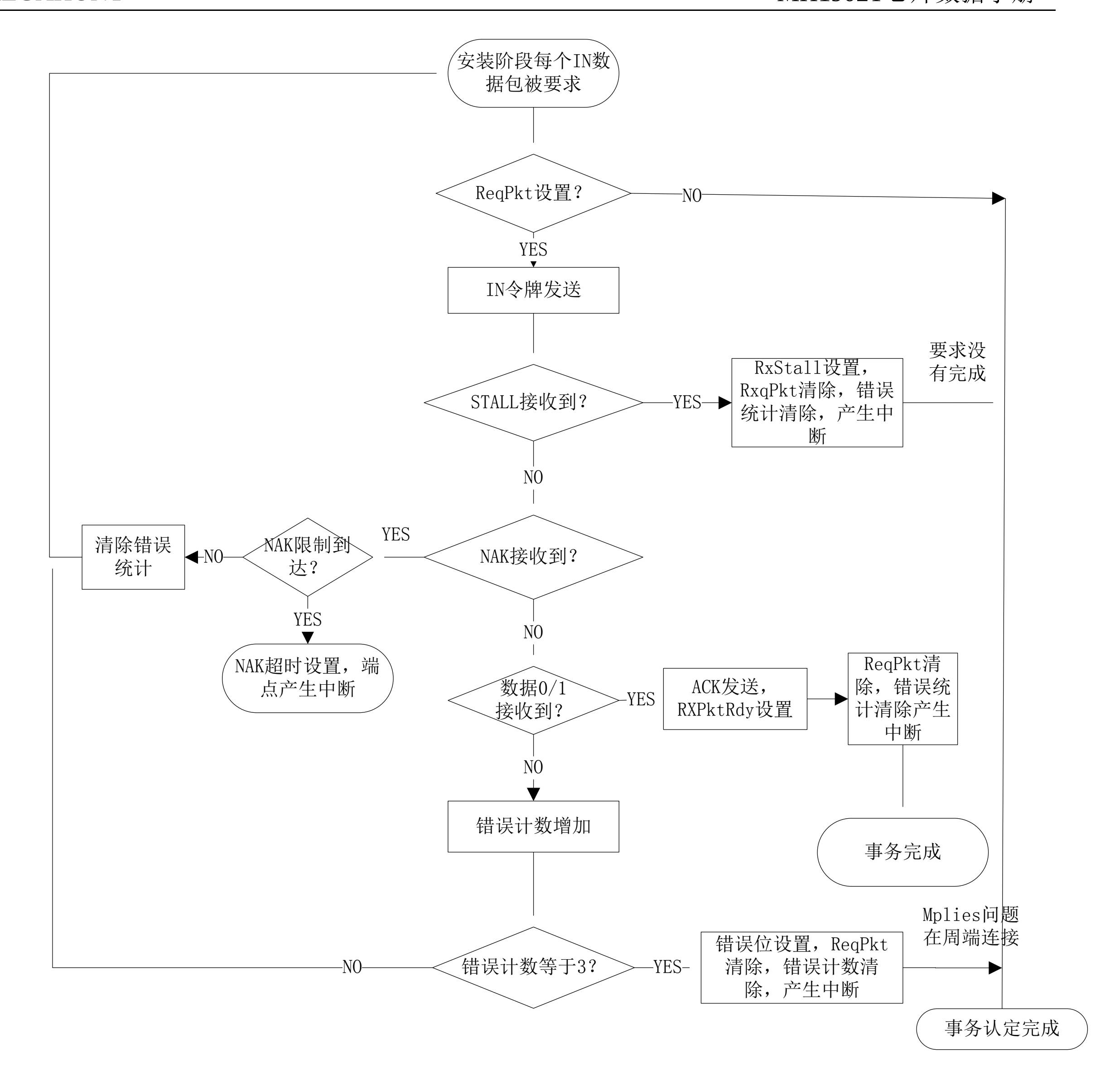
19.14.1控制事务

安装阶段

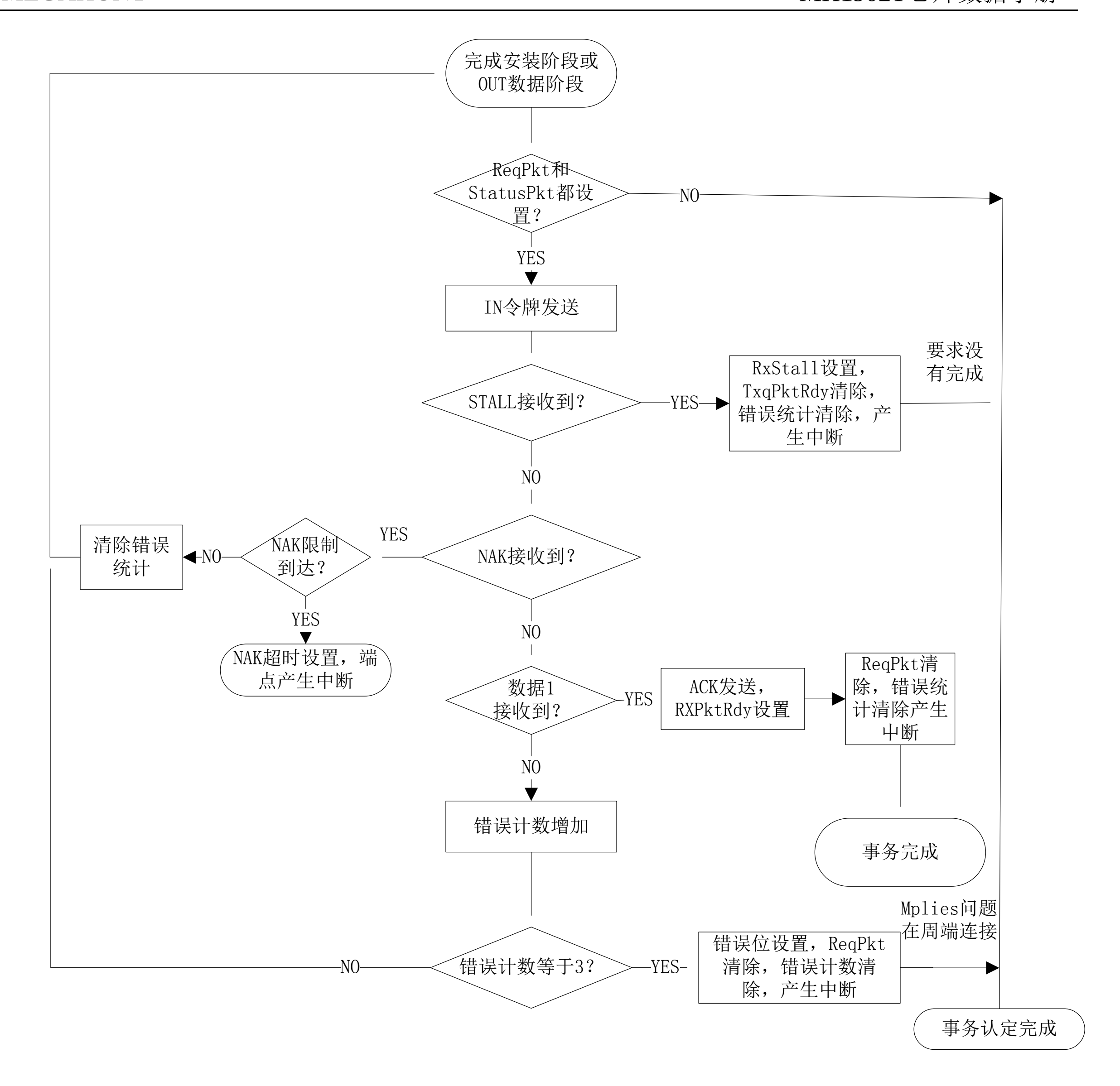


计划的事务
TxPktRdy和SetupPkt都设置？
NO
YES
STALL令牌发送
数据0发送
STALL接收到？
YES
RxStall设置，TxPktRdy清除，错误统计清除，产生中断
命令不通过目标支持
NO
ACK接收到？
YES
TxPktRdy清除，错误统计清除产生中断
NO
事务完成
清除错误统计
NO
NAK限制到达？
YES
NAK接收到？
NO
YES
错误计数增加
NAK超时设置，端点产生中断
NO
错误计数等于3？
YES
错误位设置，ReqPkt清除，错误计数清除，产生中断
Mplies问题在周端连接
事务认定完成

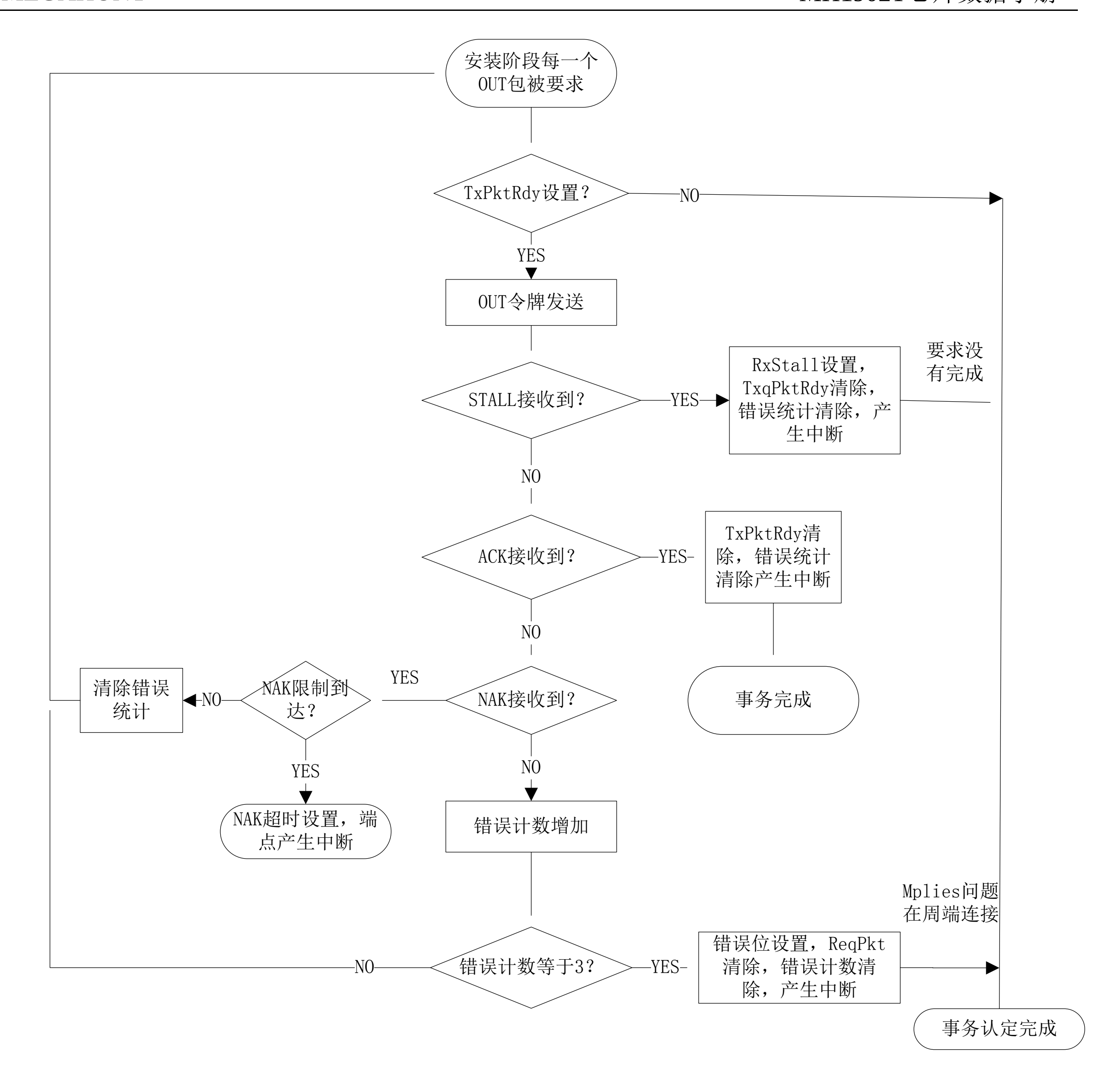
IN数据阶段



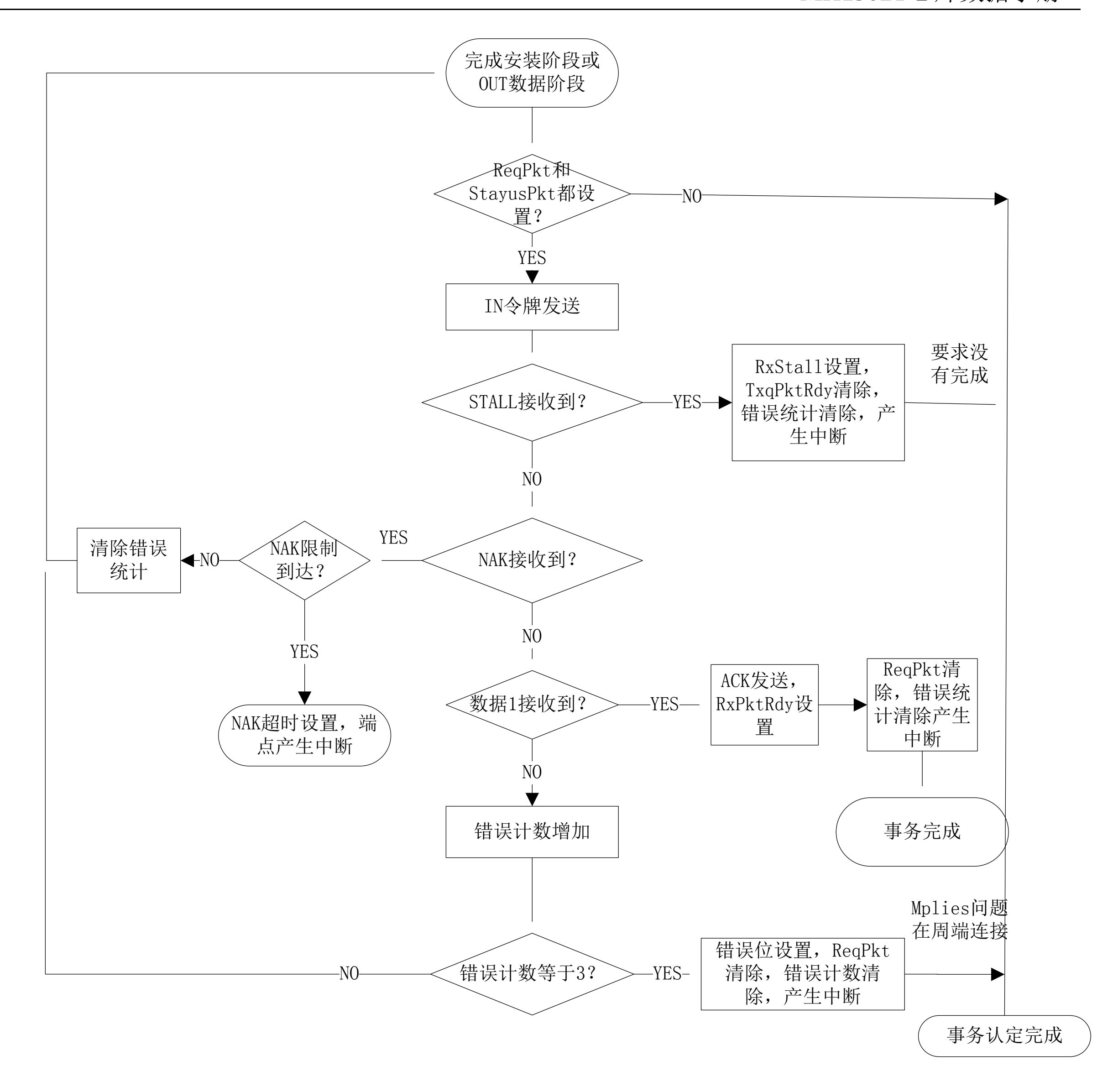
Following状态阶段


完成安装阶段或
OUT数据阶段
ReqPkt和
StatusPkt都设
置？
NO
YES
IN令牌发送
STALL接收到？
YES
RxStall设置，
TxqPktRdy清除，
错误统计清除，产
生中断
要求没
有完成
NO
清除错误
统计
NO
NAK限制
到达？
YES
NAK接收到？
YES
NAK超时设置，端
点产生中断
NO
数据1
接收到？
YES
ACK发送，
RXPktRdy设置
ReqPkt清
除，错误统
计清除产生
中断
NO
错误计数增加
事务完成
Mplies问题
在周端连接
NO
错误计数等于3？
YES
错误位设置，ReqPkt
清除，错误计数清
除，产生中断
事务认定完成

OUT数据阶段

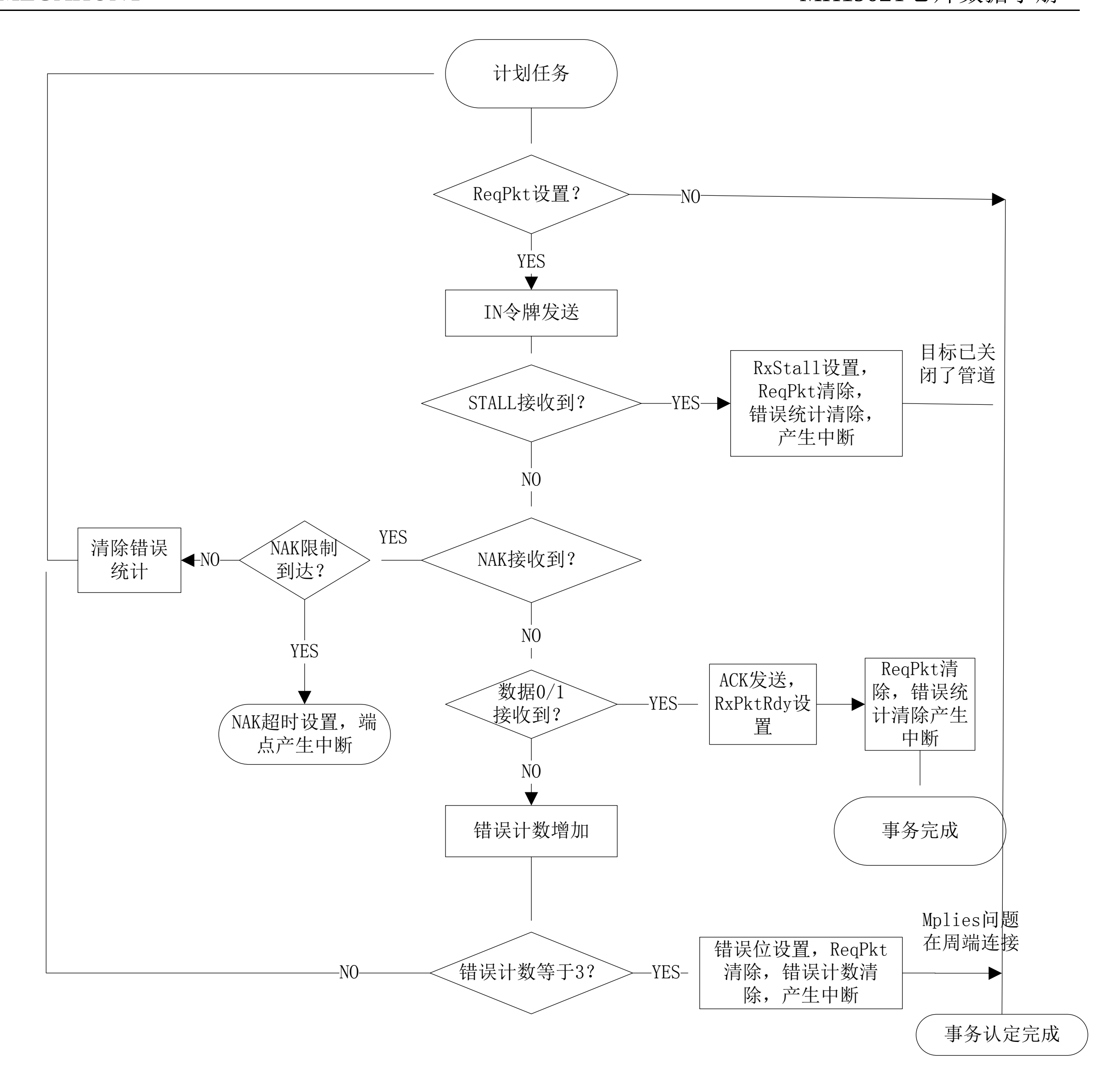


Following状态阶段

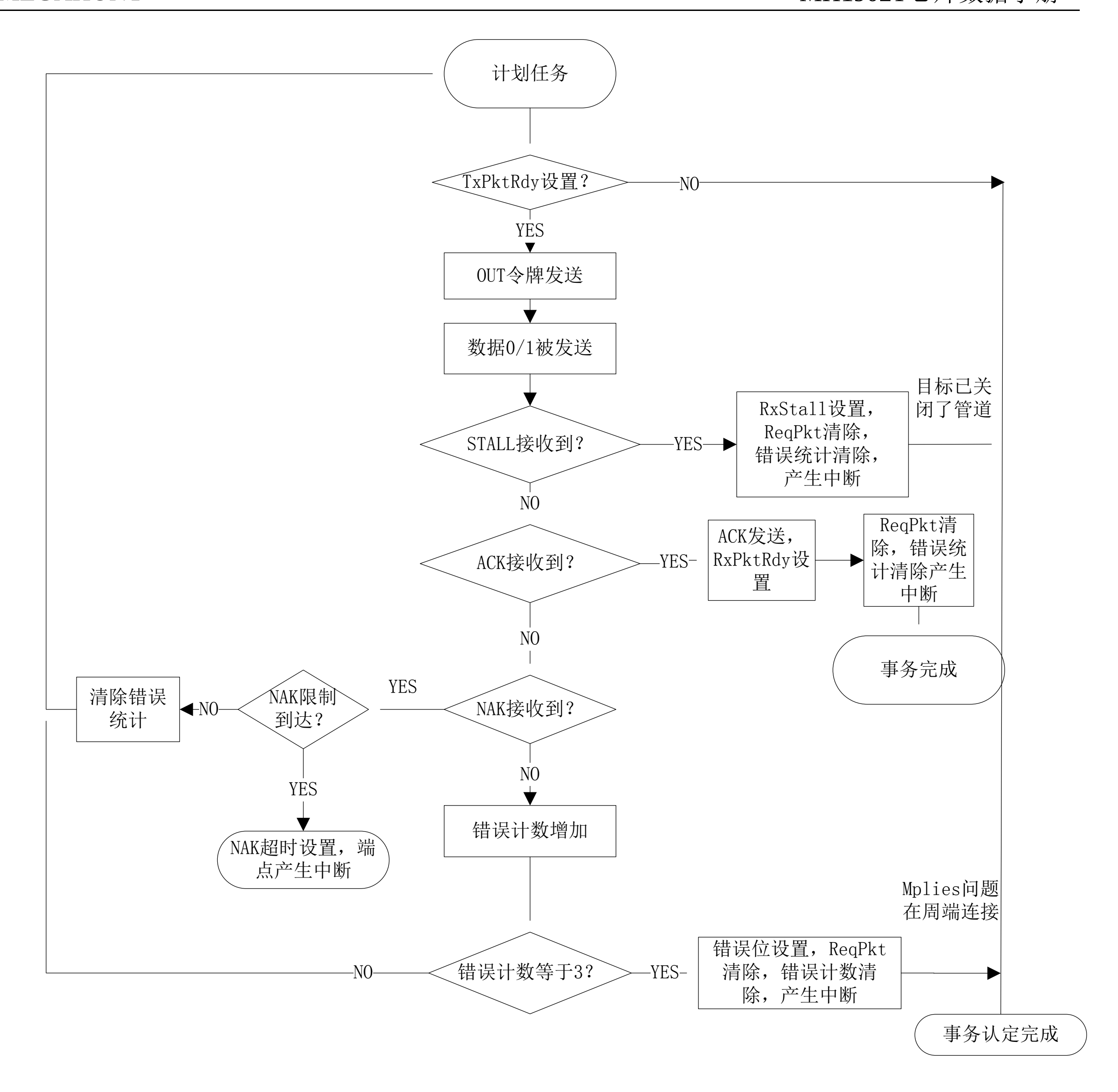


19.14.2BULK/低带宽中断事务

IN事务

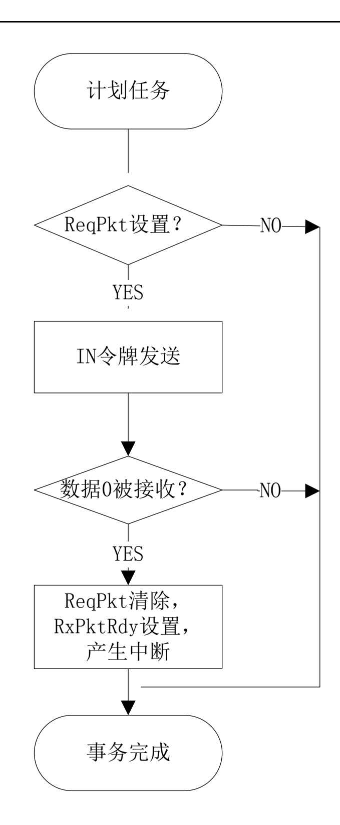

计划任务
ReqPkt设置？
NO
YES
IN令牌发送
STALL接收到？
YES
RxStall设置，
ReqPkt清除，
错误统计清除，
产生中断
目标已关
闭了管道
NO
清除错误
统计
NO
NAK限制
到达？
YES
NAK接收到？
YES
NAK超时设置，端
点产生中断
NO
数据0/1
接收到？
YES
ACK发送，
RxPktRdy设
置
ReqPkt清
除，错误统
计清除产生
中断
NO
错误计数增加
事务完成
Mplies问题
在周端连接
NO
错误计数等于3？
YES
错误位设置，ReqPkt
清除，错误计数清
除，产生中断
事务认定完成

OUT事务

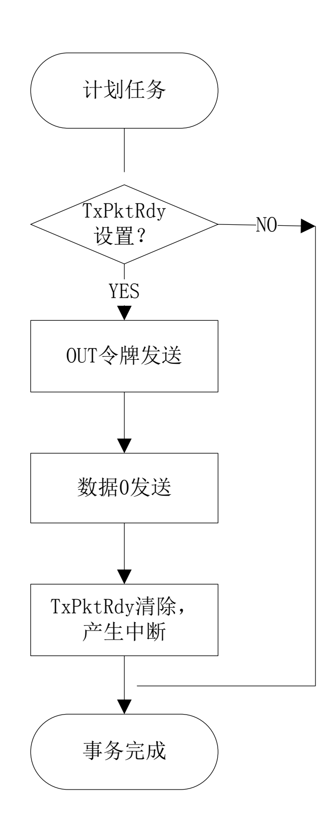


19.14.3全速/低带宽中断事务

IN事务

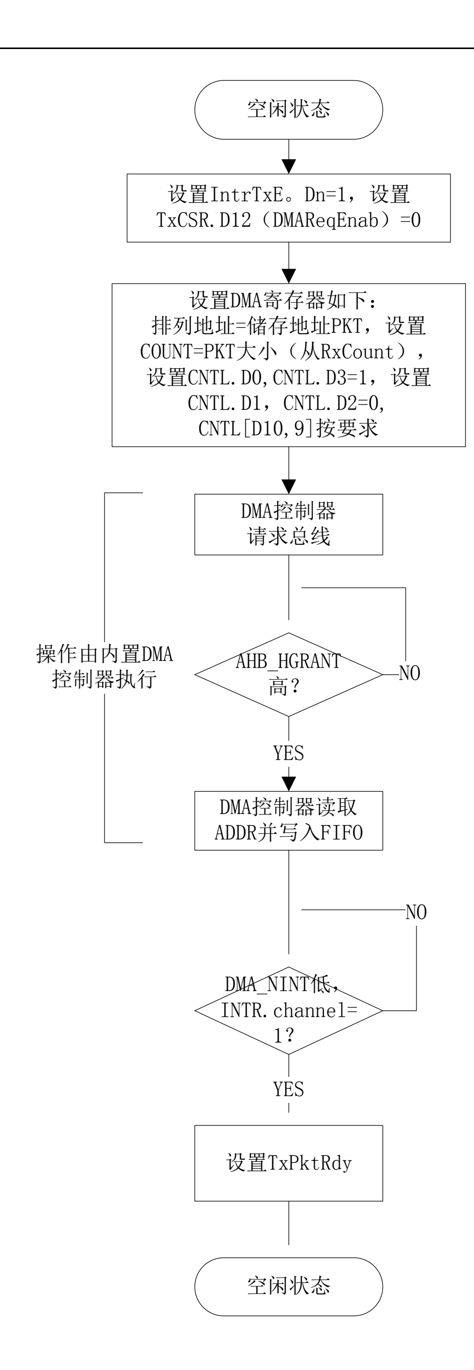


OUT事务

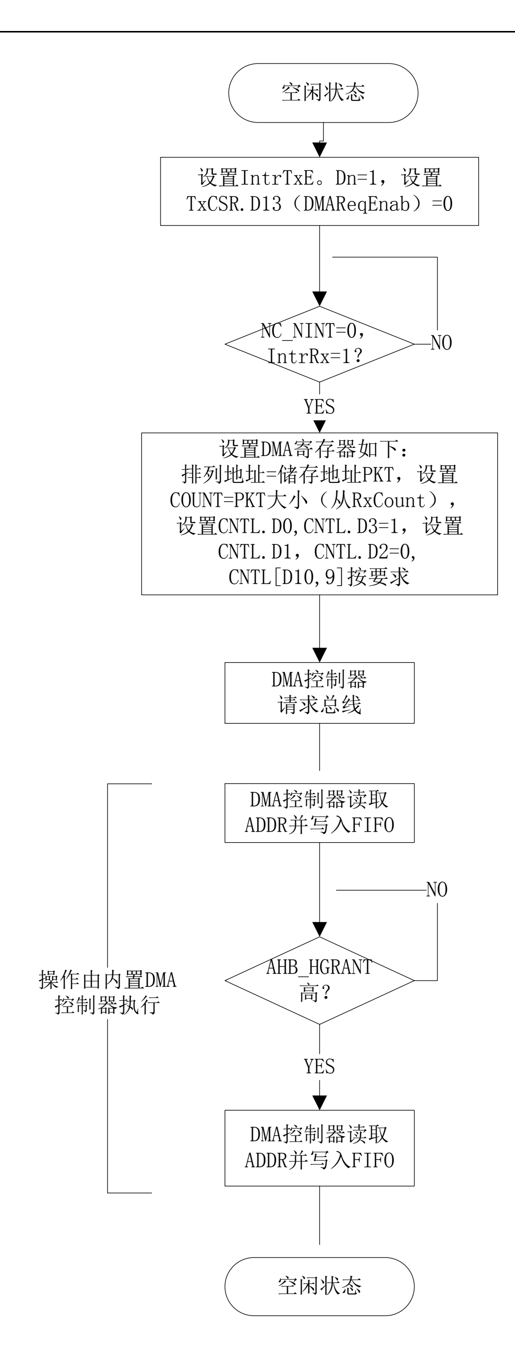


19.14.4DMA操作（与内置DMA控制器）

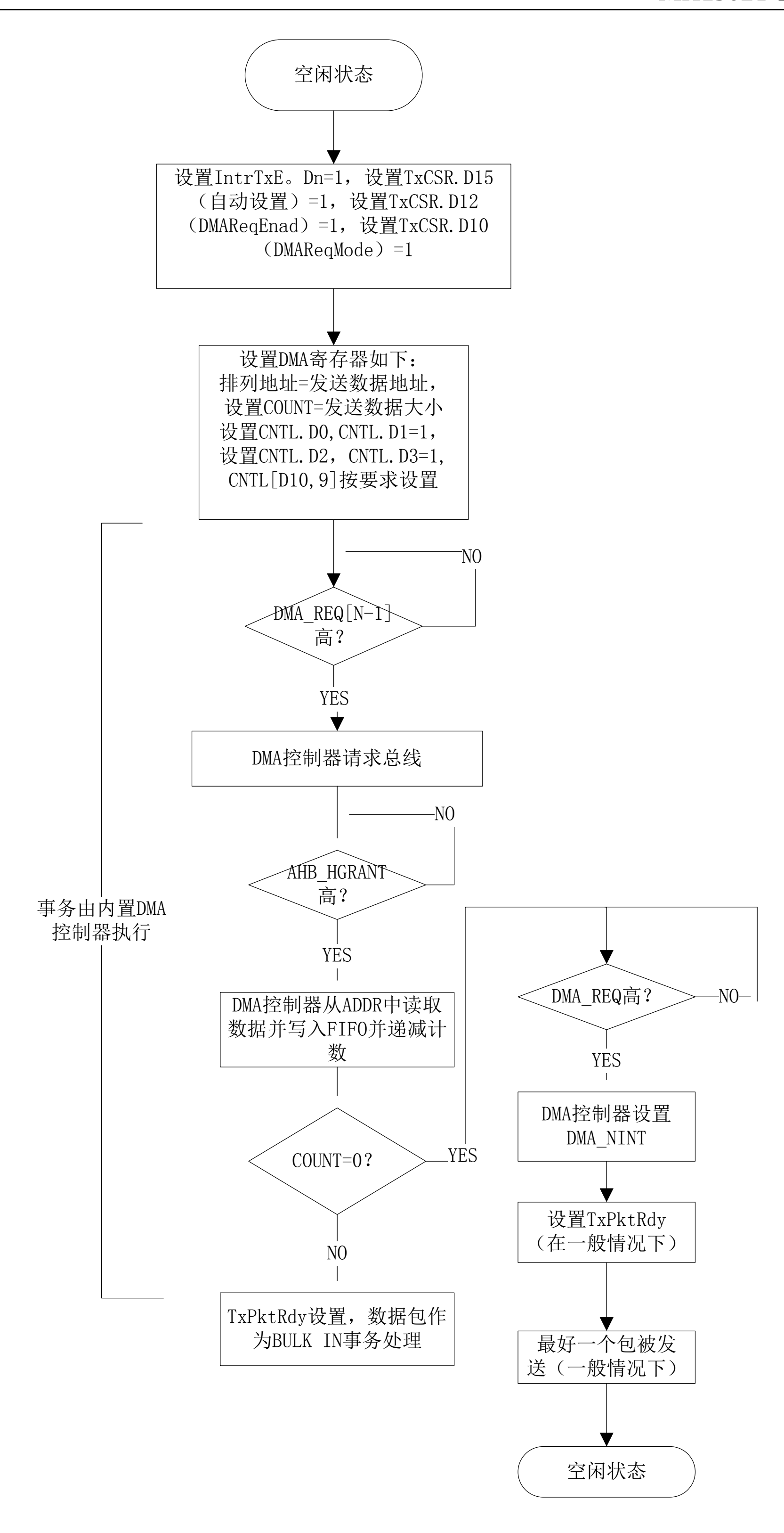
单个数据包TX


空闲状态
设置IntrTxE。Dn=1，设置
TxCSR.D12（DMAReqEnab）=0
设置DMA寄存器如下：
排列地址=储存地址PKT，设置
COUNT=PKT大小（从RxCount），
设置CNTL.D0,CNTL.D3=1，设置
CNTL.D1，CNTL.D2=0,
CNTL[D10,9]按要求
DMA控制器
请求总线
操作由内置DMA
控制器执行
AHB_HGRANT
高？
NO
YES
DMA控制器读取
ADDR并写入FIFO
NO
DMA_NINT低，
INTR.channel=
1？
YES
设置TxPktRdy
空闲状态

单个数据包RX

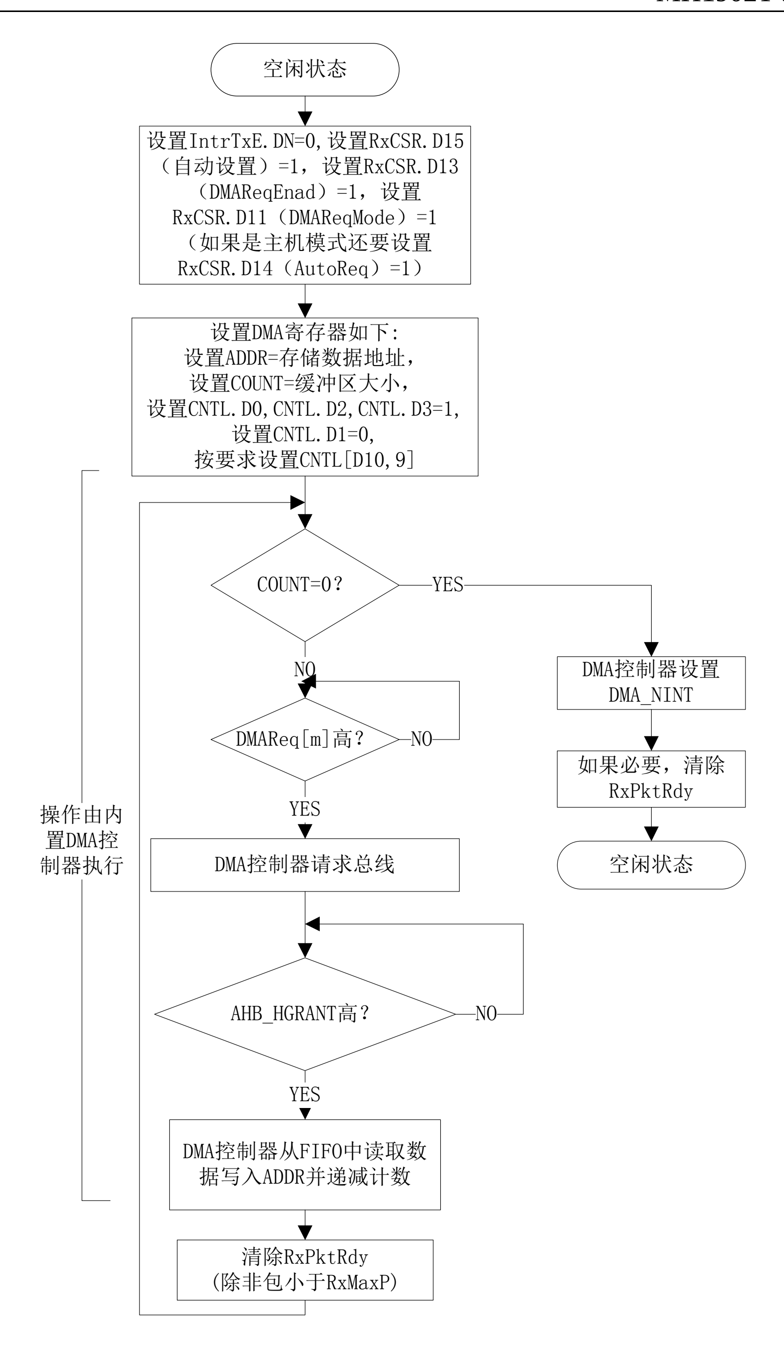

空闲状态
设置IntrTxE。Dn=1，设置
TxCSR.D13（DMAReqEnab）=0
NC_NINT=0，
IntrRx=1？
NO
YES
设置DMA寄存器如下：
排列地址=储存地址PKT，设置
COUNT=PKT大小（从RxCount），
设置CNTL.D0,CNTL.D3=1，设置
CNTL.D1，CNTL.D2=0,
CNTL[D10,9]按要求
DMA控制器
请求总线
操作由内置DMA
控制器执行
DMA控制器读取
ADDR并写入FIFO
NO
AHB_HGRANT
高？
YES
DMA控制器读取
ADDR并写入FIFO
空闲状态

多个数据包TX

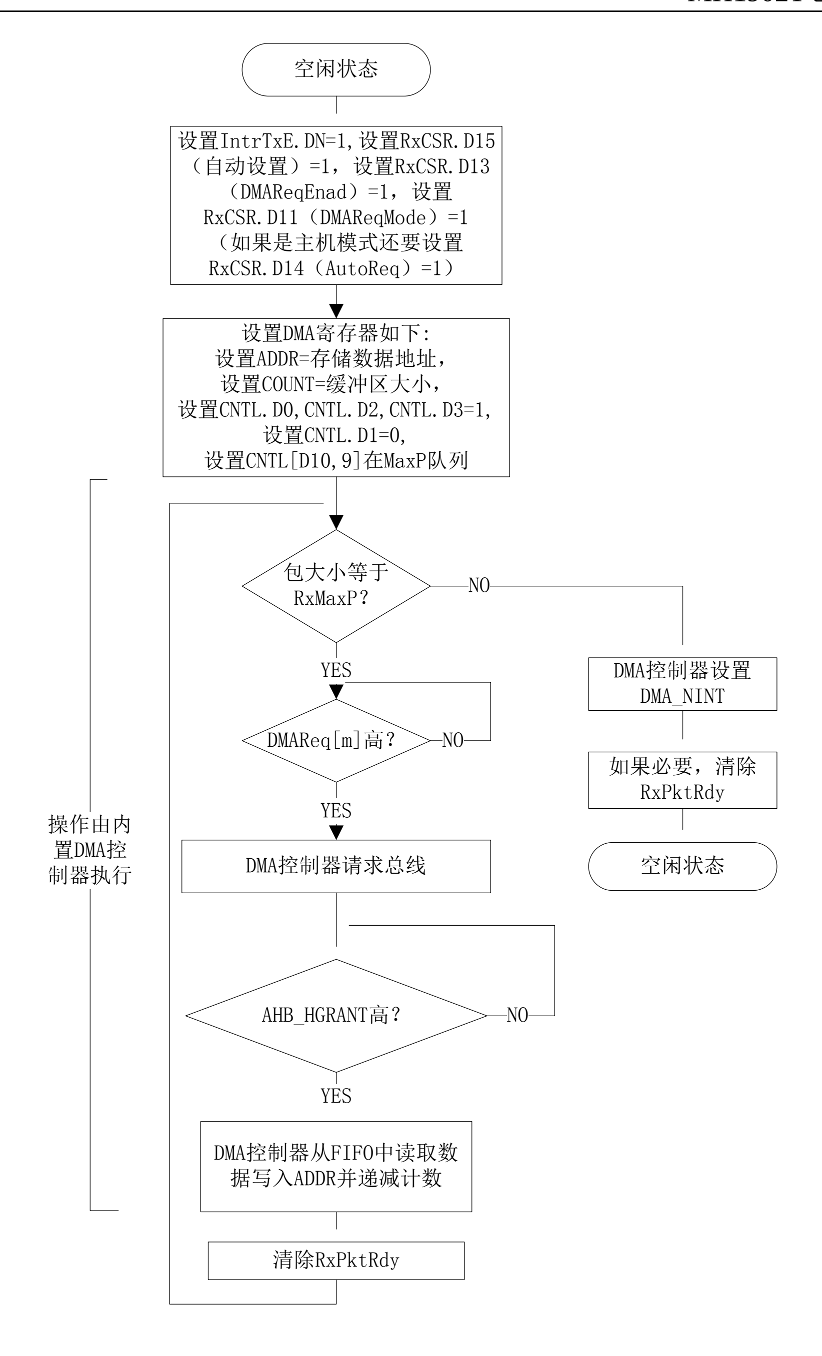


多个数据包RX

如果已知数据块大小：


空闲状态
设置IntrTxE.DN=0,设置RxCSR.D15（自动设置）=1，设置RxCSR.D13（DMAReqEnad）=1，设置RxCSR.D11（DMAReqMode）=1（如果是主机模式还要设置RxCSR.D14（AutoReq）=1）
设置DMA寄存器如下:
设置ADDR=存储数据地址，
设置COUNT=缓冲区大小，
设置CNTL.D0,CNTL.D2,CNTL.D3=1,
设置CNTL.D1=0,
按要求设置CNTL[D10,9]
COUNT=0？
YES
NO
DMA控制器设置DMA_NINT
如果必要，清除RxPktRdy
空闲状态
DMAReq[m]高？
NO
YES
操作由内置DMA控制器执行
DMA控制器请求总线
AHB_HGRANT高？
NO
YES
DMA控制器从FIFO中读取数据写入ADDR并递减计数
清除RxPktRdy
(除非包小于RxMaxP)

如果数据块大小未知：


空闲状态
设置IntrTxE.DN=1,设置RxCSR.D15（自动设置）=1，设置RxCSR.D13（DMAReqEnad）=1，设置RxCSR.D11（DMAReqMode）=1（如果是主机模式还要设置RxCSR.D14（AutoReq）=1）
设置DMA寄存器如下:
设置ADDR=存储数据地址，
设置COUNT=缓冲区大小，
设置CNTL.D0,CNTL.D2,CNTL.D3=1,
设置CNTL.D1=0,
设置CNTL[D10,9]在MaxP队列
操作由内置DMA控制器执行
包大小等于RxMaxP？
NO
YES
DMA控制器设置DMA_NINT
如果必要，清除RxPktRdy
空闲状态
DMAReq[m]高？
NO
YES
DMA控制器请求总线
AHB_HGRANT高？
NO
YES
DMA控制器从FIFO中读取数据写入ADDR并递减计数
清除RxPktRdy

20 模拟/数字转换（ADC）

20.1 ADC简介

ADC采样率为1MHz，采样精度为10比特

20.2 ADC特性

ADC有7个通道，最高采样率为1MHz，采样精度为10bit，其中ADC通道0采集电压范围为0~5V(内部分压电阻1.7M/425K)，通道6内部固定采集CHARGE_VBAT电压；其他通道采集电压范围为0~1.2V

20.3 ADC寄存器

20.3.1 地址映射表

ADC基地址表

地址范围	基地址	外设	总线
0x4001_4000-0x4001_4FFF	0x4001_4000	ADC	APB0

表 20-1 ADC寄存器表

偏移地址	寄存器名称	宽度（bit）	复位值
0x0000	ADC_CR1	32	0x00000107
0x0004	ADC_SR	32	0x00000000
0x0008	ADC_FIFO	32	0x00000000
0x000C	ADC_DATA	32	0x00000000
0x0010	ADC_FIFO_FL	32	0x00000000
0x0014	ADC_FIFO_THR	32	0x00000000
0x0018	ADC_CR2	32	0x 0000F020

20.3.2 ADC控制寄存器1（ADC_CR1）

偏移地址：0x0000 复位值：0x00000107

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
预留															
ro															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
预留							pd_adc	预留	samp_en	adc_ie	adc_samp_rate		adc_ch_sel		
ro							rw	ro	rw	rw	rw		rw		

Bit	Name	W/R	Description
31:9	预留	--	保留位
8	pd_adc	RW	ADC power done enable 0：启动ADC 1：关闭ADC
7	预留	--	保留位
6	samp_en	RW	ADC采样使能
5	adc_ie	W/R	ADC中断使能： 0：不使能； 1：使能

4:3	adc_samp_rate	W/R	ADC速率选择： 2'b00: 1M；2'b01: 500K； 2'b10: 250K；2'b11: 125K；
2:0	adc_ch_sel	W/R	ADC通道选择： 3'b000：通道1； 3'b001：通道2； 3'b010：通道3； 3'b011：通道4； 3'b100：通道5； 3'b101：通道6； 3'b110：通道7；

20.3.3 ADC状态寄存器（ADC_SR）

偏移地址：0x0004 复位值：0x00000400

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
预留															
ro															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
预留														fifo_ov	adc_done_flag
ro														rw	rw

Bit	Name	W/R	Description
31:2	预留	--	保留位
1	fifo_ov	RC	fifo溢出中断，表示空且读 或 满且写
0	adc_done_flag	RO	ADC操作完成标识： 未启用FIFO，读清 1：已完成；0：未完成 启用FIFO，只有当FIFO数据个数小于门限，才清0 1：FIFO数据个数大于门限 0：FIFO数据个数小于等于门限

20.3.4 ADC FIFO控制寄存器（ADC_FIFO）

偏移地址：0x0008 复位值：0x00000000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
预留															
ro															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
预留													fifo_ov_ie	fifo_rst	fifo_en
ro													rw	wc	rw

Bit	Name	W/R	Description
31:3	预留	--	保留位
2	fifo_ov_ie	RW	fifo溢出中断使能，高有效
1	fifo_rst	WC	置1后重置FIFO，软件置1，硬件清0
0	fifo_en	RW	ADC FIFO使能，高有效

20.3.5 ADC数据寄存器（ADC_DATA）

偏移地址：0x000C　　复位值：0x00000000

<table>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td colspan="16">预留</td></tr>
<tr><td colspan="16">ro</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="6">预留</td><td colspan="10">adc_data</td></tr>
<tr><td colspan="6">ro</td><td colspan="10">ro</td></tr>
</table>

Bit	Name	W/R	Description
31:10	预留	--	预留
9:0	adc_data	RO	ADC数据，FIFO使能后为FIFO入口

20.3.6 ADC FIFO Level寄存器（ADC_FIFO_FL）

偏移地址：0x0010　　复位值：0x00000000

<table>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td colspan="16">预留</td></tr>
<tr><td colspan="16">ro</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="11">预留</td><td colspan="5">fl</td></tr>
</table>

Bit	Name	W/R	Description
31:5	预留	--	预留
4:0	fl	RO	反映写操作FIFO中数据个数

20.3.7 ADC FIFO门限设置寄存器（ADC_FIFO_THR）

偏移地址：0x0014　　复位值：0x00000000

<table>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td colspan="16">预留</td></tr>
<tr><td colspan="16">ro</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="11">预留</td><td colspan="5">thr</td></tr>
<tr><td colspan="11">ro</td><td colspan="5">ro</td></tr>
</table>

Bit	Name	W/R	Description
31:5	预留	--	
4:0	thr	RO	FIFO门限值 0：有1个以上数据产生中断 1：有2个以上数据产生中断 2：有3个以上数据产生中断 x：有x+1个以上数据产生中断

21 数字/模拟转换（DAC）

21.1 DAC简介

DAC是10位数字输入、电压输出的数字/模拟转换器。

21.2 DAC主要特性

- 1 个通道
- 10 位数字输入
- 支持 DMA 功能

21.3 DAC寄存器

21.3.1 地址映射表

DAC基地址表

地址范围	基地址	外设	总线
0x4001_4100-0x4001_41FF	0x4001_4100	DAC	APB0

表 21-1 DAC寄存器表

偏移地址	寄存器名称	宽度（bit）	复位值
0x0000	DAC_CR1	32	0x00000010
0x0004	DAC_DATA	32	0x00000000
0x0008	DAC_TIMER	32	0x00000000
0x000C	DAC_FIFO_FL	32	0x00000000
0x0010	DAC_FIFO_THR	32	0x00000000

21.3.2 DAC控制寄存器（DAC_CR1）

偏移地址：0x0000　　复位值：0x00000010

31	30	29	28	27	26	25	24
fifo_is	fifo_ov	dac_runing	预留				

23	22	21	20	19	18	17	16
预留							

15	14	13	12	11	10	9	8
预留							

7	6	5	4	3	2	1	0
预留		dac_curr_sel	pd_dac	fifo_rst	dac_dma_en	is_ie	ov_ie

Bit	Name	W/R	Description
31	fifo_is	RO	DAC FIFO门限中断标志位，硬件清0 1：DAC FIFO中的个数小于等于门限值 0：DAC FIFO中的个数大于门限值
30	fifo_ov	RC	fifo溢出中断，表示空且读 或 满且写，读清清除该标志位

29	dac_runing	RO	DAC运行标志，最后一个数据从FIFO读取后，需要经过一段时间后，DAC才能发送完成，所以可通过该位确认是否发送完成 1：DAC运行中 0：DAC停止
28:6	预留	RO	保留位，读取值为0
5	dac_curr_sel	RW	DAC工作模式选择： 0:VOUTP接至少20K电阻 1:VOUTP接2K电阻
4	pd_dac	RW	开启ADC或DAC，需将PD_ADC和PD_DAC均清0 0：打开DAC 1：关闭DAC
3	fifo_rst	WC	置1后重置FIFO，软件置1，硬件清0
2	dac_dma_en	W/R	DAC DMA使能，置1使能DMA在计数器与期望值相等时启动DMA搬运数据至DAC
1	is_ie	W/R	DAC FIFO中断使能，置1使能FIFO门限中断（不影响fifo_is标志位置1）
0	ov_ie	W/R	FIFO溢出中断使能，置1使能FIFO溢出中断（不影响fifo_ov标志位置1）

21.3.3 DAC数据寄存器（DAC_DATA）

偏移地址：0x0004 复位值：0x00000000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
预留															
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
预留						dac_data									

Bit	Name	W/R	Description
31:10	预留	RO	保留
9:0	dac_data	RO	DAC数据寄存器，FIFO入口

21.3.4 DAC定时器（DAC_TIMER）

偏移地址：0x0008 复位值：0x00000000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
dac_timer															
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
dac_timer															

Bit	Name	W/R	Description
31:0	dac_timer	RW	DAC定时器期望值，每计(dac_timer+1)*2 个PCLK便从DAC FIFO中获取数据启动转换

21.3.5 DAC FIFO Level寄存器（DAC_FIFO_FL）

偏移地址：0x000C 复位值：0x00000000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
预留															
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**

预留	fl

Bit	Name	W/R	Description
31:4	预留	RO	保留
3:0	fl	RO	反映写操作FIFO中数据个数

21.3.6 DAC FIFO门限设置寄存器（DAC_FIFO_THR）

偏移地址：0x0014　　复位值：0x00000000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
预留															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
预留												thr			

Bit	Name	W/R	Description
31:4	预留	RO	保留
3:0	thr	RO	FIFO门限值 FIFO数据个数小于等于该门限请求DMA或产生中断要求CPU写入数据，大于该门限结束请求

22 QSPI控制器（QFlash_controller）

22.1 QSPI控制器简介

芯片提供可串行访问外部FLASH器件的控制器。QSPI控制器支持标准SPI，Dual SPI以及Quad SPI通讯。

22.2 QSPI控制器功能特性

- 支持单线（Single），两线（Double）和四线（Quad）I/O 命令
- 最高支持与 HCLK 时钟同频率
- 可配置 DMA 控制器进行访问
- 命令参数/格式、SPI 极性/相位等均可配置，具有良好的兼容性

22.3 QSPI控制器寄存器

22.3.1 地址映射表

寄存器基地址表

地址范围	基地址	外设	总线
0x4005_0000-0x4005_FFFF	0x4005_0000	QFlash_controller	AHB

表 22-1 QSPI控制器寄存器表

偏移地址	寄存器名称	宽度（bit）	复位值
0x00	FCU_CMD	32	0x00000000
0x04	ADDRESS	32	0x00000000
0x08	BYTE_NUM	32	0x00000000
0x0C	WR_FIFO	32	0x00000000
0x10	RD_FIFO	32	0x00000000
0x14	DEVICE_PARA	32	0x00480083
0x18	REG_WDATA	32	0x00000000
0x1C	REG_RDATA	32	0x00000000
0x20	INT_MASK	32	0x00000000
0x24	INT_UMMASK	32	0x00000000
0x28	INT_MASK_STATUS	32	0x00000000
0x2C	INT_STATUS	32	0x00000000
0x30	INT_RAWSTATUS	32	0x00000000
0x34	INT_CLEAR	32	0x00000000
0x38	CACHE_INTF_CMD	32	0xABB92BEB
0x3C	DMA_CNTL	32	0x00000000
0x40	FIFO_CNTL	32	0x00000000

22.3.2 命令寄存器（FCU_CMD）

偏移地址：0x0000　　复位值：0x00000000

31	30	29	28	27	26	25	24
command_code							
23	22	21	20	19	18	17	16
预留							
15	14	13	12	11	10	9	8

预留						bus_mode	
7	6	5	4	3	2	1	0
cmd_format				done	busy	access_ack	access_req

Bit	Name	W/R	Description
31:24	command_code	RW	命令字
23:10	预留	--	
9:8	bus_mode	RW	00：SPI mode，cmd-addr-data = 1-1-1 01：Q-out mode，cmd-addr-data = 1-1-4 10：Q-IO mode, cmd-addr-data = 1-4-4 11：QPI mode，cmd-addr-data = 4-4-4
7:4	cmd_format	RW	命令格式： 0000: 8bit cmd 0001: 8bit cmd + 8bit read reg data 0010: 8bit cmd + 16bit read reg data 0011: 8bit cmd + 24bit read reg data 0100: 8bit cmd + 24bit dummy + 8bit write reg data 0101: 8bit cmd + 24bit地址（release from power down,地址设置为0） + 8bit read reg data 0110: 8bit cmd + 24bit地址（release from power down,地址设置为0） + 16bit read reg data 0111: 8bit cmd + 8bit write reg data 1000: 8bit cmd + 16bit write reg data 1001: 8bit cmd + 24bit 地址 1010: 8bit cmd + 24bit addr + 读data… 1011: 8bit cmd + 24bit addr + dummy +读data... 1100: 8bit cmd + 24bit addr + 8bitM + dummy + 读data... 1101: 8bit cmd + 24bit addr + 编程data...
3	done	RO	当前命令完成标志
2	busy	RO	当前命令正在进行
1	access_ack	RO	1：当前命令requst被许可 0：当前code bus正在访问，不许可s-bus访问Flash
0	access_req	RW	请求访问Flash

22.3.3 地址寄存器（ADDRESS）

偏移地址：0x0004 复位值：0x00000000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
address															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
address								m7_0							

Bit	Name	W/R	Description
31:8	address	RW	命令执行的Flash地址
7:0	m7_0	RW	m7~m0：是否做连续读（M5-M4=(10))

22.3.4 读取数据数量寄存器（BYTE_NUM）

偏移地址：0x0008　　　　　　　　　　复位值：0x00000000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留			wr_byte_num												

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留			rd_byte_num												

Bit	Name	W/R	Description
31:29	保留	--	
28:16	wr_byte_num	RW	写命令后的写数据字节数量， 1-4096字节，写‘0’表示0，写"1_0000_0000"表示4096
15:13	保留	RO	
12:0	rd_byte_num	RW	读命令后的读数据字节数量， 1-4096字节，写‘0’表示0，写"1_0000_0000"表示4096

22.3.5 写数据FIFO寄存器（WR_FIFO）

偏移地址：0x000C　　　　　　　　　　复位值：0x00000000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
wr_data															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
wr_data															

Bit	Name	W/R	Description
31:0	wr_data	WO	写数据FIFO，32位，按wr_byte_num的实际数据发送到Flash

22.3.6 读数据FIFO寄存器（RD_FIFO）

偏移地址：0x0010　　　　　　　　　　复位值：0x00000000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
rd_data															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
rd_data															

Bit	Name	W/R	Description
31:0	rd_data	RO	读数据FIFO，32位， 如果rd_byte_num不是4的整数倍，LSB补0

22.3.7 器件参数寄存器（DEVICE_PARA）

偏移地址：0x0014　　　　　　　　　　复位值：0x00480083

31	30	29	28	27	26	25	24
one_us_count							

23	22	21	20	19	18	17	16
one_us_count							

15	14	13	12	11	10	9	8
sample_dly	sample_pha	预留				protocol	

7	6	5	4	3	2	1	0
dummy_cycles				flash_ready	预留	freq_sel	

Bit	Name	W/R	Description
31:16	one_us_count	RW	以controller时钟周期计数1us的数值
15	sample_dly	RW	采样延迟时间： 0：延迟1个周期采样返回的Flash读数据 1：延迟2个周期采样返回的Flash读数据
14	sample_pha	RW	采样时钟相位选择： 0：使用non-inverted时钟（非反相时钟）的上升沿采样Flash读数据 1：使用inverted时钟（反相时钟）的上升沿采样Flash读数据
13:10	保留	--	
9:8	protocol	RW	00：CLPL（clock inactive low） 11：CHPH（clock inactive high） 01,10：reserved
7:4	dummy_cycles	RW	dummy cycle的数量包括了24位地址后面的8bit M-code
3	flash_ready	RW	0: Flash器件初始化未完成，Cache读操作被暂停 1：Flash器件初始化完成，Cache可以读Flash
2	保留	--	
1:0	freq_sel	RW	00: qspi_clk = fclk 01: qspi_clk = fclk/2 10: qspi_clk = fclk/3 11: qspi_clk = fclk/4

22.3.8 FLASH寄存器写数据（REG_WDATA）

偏移地址：0x0018　　复位值：0x00000000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
reg_wdata															
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
reg_wdata															

Bit	Name	W/R	Description
31:0	reg_wdata	RW	写寄存器数据

22.3.9 FLASH寄存器读数据（REG_RDATA）

偏移地址：0x001C　　复位值：0x00000000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
reg_rdata															
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
reg_rdata															

Bit	Name	W/R	Description
31:0	reg_rdata	WO	读寄存器数据

22.3.10中断MASK寄存器（INT_MASK）

偏移地址：0x0020 复位值：0x00000000

31	30	29	28	27	26	25	24
预留							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
预留							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
预留							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
预留	tx_fifo_data_mask	rx_fifo_data_mask	tx_fifo_of_mask	tx_fifo_uf_mask	rx_fifo_of_mask	rx_fifo_uf_mask	done_int_mask

Bit	Name	W/R	Description
31:7	保留	RO	
6	tx_fifo_data_mask	RW	写1不使能tx FIFO的数据中断
5	rx_fifo_data_mask	RW	写1不使能rx FIFO的数据中断
4	tx_fifo_of_mask	RW	写1不使能tx FIFO overflow中断
3	tx_fifo_uf_mask	RW	写1不使能tx FIFO underflow中断
2	rx_fifo_of_mask	RW	写1不使能rx FIFO overflow中断
1	rx_fifo_uf_mask	RW	写1不使能rx FIFO underflow中断
0	done_int_mask	RW	写1不使能命令完成中断

22.3.11中断UNMASK寄存器（INT_UNMASK）

偏移地址：0x0024 复位值：0x00000000

31	30	29	28	27	26	25	24
预留							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
预留							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
预留							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
预留	tx_fifo_data_unmask	rx_fifo_data_unmask	tx_fifo_of_unmask	tx_fifo_uf_unmask	rx_fifo_of_unmask	rx_fifo_uf_unmask	done_int_unmask

Bit	Name	W/R	Description
31:7	保留	--	
6	tx_fifo_data_unmask	RW	写1使能tx FIFO的数据中断
5	rx_fifo_data_unmask	RW	写1使能rx FIFO的数据中断
4	tx_fifo_of_unmask	RW	写1不使能tx FIFO overflow中断
3	tx_fifo_uf_unmask	RW	写1不使能tx FIFO underflow中断
2	rx_fifo_of_unmask	RW	写1不使能rx FIFO overflow中断
1	rx_fifo_uf_unmask	RW	写1不使能rx FIFO underflow中断
0	done_int_unmask	RW	写1不使能命令完成中断

22.3.12中断MASK状态寄存器（INT_MASK_STATUS）

偏移地址：0x0028 复位值：0x00000000

31	30	29	28	27	26	25	24

23	22	21	20	19	18	17	16
预留							
15	14	13	12	11	10	9	8
预留							
7	6	5	4	3	2	1	0
预留	tx_fifo_data_mask	rx_fifo_data_mask	tx_fifo_of_mask	tx_fifo_uf_mask	rx_fifo_of_mask	rx_fifo_uf_mask	done_int_mask

Bit	Name	W/R	Description
31:7	预留	--	
6	tx_fifo_data_mask	RO	tx FIFO的数据中断使能状态
5	rx_fifo_data_mask	RO	rx FIFO的数据中断使能状态
4	tx_fifo_of_mask	RO	tx FIFO overflow中断使能状态
3	tx_fifo_uf_mask	RO	tx FIFO underflow中断使能状态
2	rx_fifo_of_mask	RO	rx FIFO overflow中断使能状态
1	rx_fifo_uf_mask	RO	rx FIFO underflow中断使能状态
0	done_int_mask	RO	命令完成中断使能状态

22.3.13中断状态寄存器（INT_ STATUS）

偏移地址：0x002C　　　　复位值：0x00000000

31	30	29	28	27	26	25	24
预留							
23	22	21	20	19	18	17	16
预留							
15	14	13	12	11	10	9	8
预留							
7	6	5	4	3	2	1	0
预留	tx_fifo_data_status	rx_fifo_data_status	tx_fifo_of_status	tx_fifo_uf_status	rx_fifo_of_status	rx_fifo_uf_status	done_int_status

Bit	Name	W/R	Description
31:7	预留	RO	
6	tx_fifo_data_status	RO	tx FIFO的数据中断状态
5	rx_fifo_data_status	RO	rx FIFO的数据中断状态
4	tx_fifo_of_status	RO	tx FIFO overflow中断状态
3	tx_fifo_uf_status	RO	tx FIFO underflow中断状态
2	rx_fifo_of_status	RO	rx FIFO overflow中断状态
1	rx_fifo_uf_status	RO	rx FIFO underflow中断状态
0	done_int_status	RO	命令完成中断状态

22.3.14中断原始状态寄存器（INT_ RAWSTATUS）

偏移地址：0x0030　　　　复位值：0x00000000

31	30	29	28	27	26	25	24
预留							
23	22	21	20	19	18	17	16
预留							
15	14	13	12	11	10	9	8

预留							
7	6	5	4	3	2	1	0
预留	tx_fifo_data_rawstatus	rx_fifo_data_rawstatu	tx_fifo_of_rawstatus	tx_fifo_uf_rawstatus	rx_fifo_of_rawstatus	rx_fifo_uf_rawstatus	done_int_rawstatus

Bit	Name	W/R	Description
31:7	预留	RO	
6	tx_fifo_data_rawstatus	RO	tx FIFO的数据中断状态
5	rx_fifo_data_rawstatus	RO	rx FIFO的数据中断状态
4	tx_fifo_of_rawstatus	RO	tx FIFO overflow中断状态
3	tx_fifo_uf_rawstatus	RO	tx FIFO underflow中断状态
2	rx_fifo_of_rawstatus	RO	rx FIFO overflow中断状态
1	rx_fifo_uf_rawstatus	RO	rx FIFO underflow中断状态
0	done_int_rawstatus	RO	命令完成中断状态

22.3.15中断清除寄存器（INT_ CLEAR）

偏移地址：0x0034 复位值：0x00000000

31	30	29	28	27	26	25	24
预留							
23	22	21	20	19	18	17	16
预留							
15	14	13	12	11	10	9	8
预留							
7	6	5	4	3	2	1	0
预留	clr_tx_fifo_data	clr_rx_fifo_data	clr_tx_fifo_of	clr_tx_fifo_uf	clr_rx_fifo_of	clr_rx_fifo_uf	clr_done_int

Bit	Name	W/R	Description
31:7	预留	--	
6	clr_tx_fifo_data	RO	清除tx FIFO的数据中断
5	clr_rx_fifo_data	RO	清除rx FIFO的数据中断
4	clr_tx_fifo_of	WO	清除tx FIFO overflow中断
3	clr_tx_fifo_uf	WO	清除tx FIFO underflow中断
2	clr_rx_fifo_of	WO	清除rx FIFO overflow中断
1	clr_rx_fifo_uf	WO	清除rx FIFO underflow中断
0	clr_done_int	WO	清除命令完成中断

22.3.16CACHE接口命令寄存器（CACHE_ INTF_CMD）

偏移地址：0x0038 复位值：0xABB92BEB

31	30	29	28	27	26	25	24
cache_intf_reldscmd							
23	22	21	20	19	18	17	16
cache_intf_dscmd							
15	14	13	12	11	10	9	8
force_flash_on	预留	cache_intf_rdcmd_bus_mode		cache_intf_rdcmd_format			
7	6	5	4	3	2	1	0
cache_intf_rdcmd							

Bit	Name	W/R	Description
31:24	cache_intf_reldscmd	RW	cache接口发出Flash release from deepsleep的命令，缺省0xAB
23:16	cache_intf_dscmd	RW	cache接口发出Flash deepsleep的命令，缺省0xB9
15	force_flash_on	RW	0:系统进入deepsleep模式时cache接口发Flash_deepsleep命令，外部flash进入低功耗待机模式 1：系统进入deepsleep模式时cache接口不发Flash_deepsleep命令，外部Flash不进入低功耗待机模式
14	预留	--	
13:12	cache_intf_rdcmd_bus_mode	RW	cache读Flash和Flash Deepsleep（release from DS）的总线模式： 00：SPI mode，cmd-addr-data = 1-1-1 01：Q-out mode，cmd-addr-data = 1-1-4 10: Q-IO mode, cmd-addr-data = 1-4-4 11：QPI mode，cmd-addr-data = 4-4-4
11:8	cache_intf_rdcmd_format	RW	cache读Flash的命令格式： 1011: 8bit cmd + 24bit addr + 8bitM + dummy + 读data…
7:0	cache_intf_rdcmd	RW	cache读Flash的命令字，缺省值为0xEB 0xEB: quad-IO

22.3.17DMA控制寄存器（DMA_CNTL）

偏移地址：0x003C 复位值：0x00000000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
预留															
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
预留															tx_dma_en

Bit	Name	W/R	Description
31:1	预留	RO	
0	tx_dma_en	RW	使能TX DMA，写‘1’后如果tx FIFO容量小于4则产生dma_req

22.3.18FIFO控制寄存器（FIFO_CNTL）

偏移地址：0x0040 复位值：0x00000000

31	30	29	28	27	26	25	24
tx_fifo_flush	预留						
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
预留		tx_fifo_empty	tx_fifo_full	tx_fifo_level			
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
rx_fifo_flush	预留						
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
预留		rx_fifo_empty	rx_fifo_full	预留	rx_fifo_level		

Bit	Name	W/R	Description
31	tx_fifo_flush	WC	写‘1’清空TX FIFO
30:22	预留	--	
21	tx_fifo_empty	RO	‘1’:TX FIFO为空
20	tx_fifo_full	RO	‘1’:TX FIFO为满
19:16	tx_fifo_level	RO	TX FIFO数据量
15	rx_fifo_flush	WC	写‘1’清空RX FIFO
14:6	预留	--	
5	rx_fifo_empty	RO	1':RX FIFO为空
4	rx_fifo_full	RO	1':RX FIFO为满
3	预留	--	
2:0	rx_fifo_level	RO	RX FIFO数据量

23 充电模块（CHARGE）

23.1 充电模块简介

用于给3.7V锂电池充电

- 支持最大200mA的充电电流
- 电池充满的电压为4.15±0.05V
- 电池充满后电压降到4.05V之后将重新给电池充电

23.2 充电模块特性

涓流充电电压阈值为2.9V，低于2.9V时采用涓流(小电流)方式充电，高于2.9V时采用恒流方式充电。当电池电压高于2.9V且充电电流小于恒流的1/10时(充满的电流受外接电阻的影响，当电阻为1K时，充满的电流阈值为20mA，电阻为2K时，充满的电流阈值为10mA)认为充满，将不再给电池充电。

CHARGE_PROG管脚建议接的电阻大小为1K，该电阻的阻值和充电电流成反比。即1K时最大充电电流为200mA，2K时最大充电电流为100mA。

23.3 充电模块寄存器(CARD_RSVD)

地址：0x4001_F048　　　　复位值：0x0100021E

31	30	29	28	27	26	25	24
保留					chg_state		chg_intp_en
ro					rw		rw

23	22	21	20	19	18	17	16
chg_intp	保留						
rw	ro						

15	14	13	12	11	10	9	8
保留							
ro							

7	6	5	4	3	2	1	0
保留							
ro							

Bit	Name	W/R	Description
31:27	保留	--	保留
26:25	chg_state	RW	充电状态： 11:充满 10:恒流充电 01:涓流充电 00:欠压
24	chg_intp_en	RW	1:当chg_state不为0时上报中断 0:不上报chg_state中断，但可从chg_intp中查到状态位
23	chg_intp	RW	chg中断状态位，硬件置1，软件写0清0
22:0	保留	--	保留

23.4 充电模块参数表

表 23-1 充电模块参数

编程电阻值	消流充电电流(mA)	消流阈值(V)	恒流充电电流(mA)	充电电压(V)
1K	19±1	2.87±0.01	184±5	4.15±.0.05
2K	10±1	2.88±0.01	92.5±1.5	4.15±.0.05
4K	5±1	2.90±0.01	46.5±1.5	4.15±.0.05

24 开关机电路（POWER_KEY）

24.1 开关机电路简介

开关机电路通过控制EN_LDO5V输出高低电平，可实现芯片的硬开关机的功能。

24.2 开关机电路特性

- 开关机电路使用 CHARGE_VBAT 供电，且使用开关机功能时 VBAT33 必须在位
- 芯片未上电时 POWER_KEY 拉高 1S EN_LDO5V 输出高，芯片上电后 POWER_KEY 拉高 7S EN_LDO5V 输出低(均为硬件逻辑)
- 软件可查询 POWER_KEY 键状态(对应 PD15)，可配置 POWER_KEY 键(对应 PD15)中断使能
- PD15 读取状态与 POWER_KEY 管脚实际电平相反，即若 POWER_KEY 未按下时(低电平)，软件读取 PD15 状态为高
- 软件可控制 EN_LDO5V 管脚拉低
- 通过上述两点特性可实现软件关机
- 支持低功耗唤醒
- CHARGE_VCC 有电时 EN_LDO5V 直接输出高电平且无法拉低

备注：POWER_KEY键平时为低电平，按下拉高

24.3 开关机电路寄存器

软件实现关机控制的寄存器描述见模拟控制寄存器(SEN_ANA0)。